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
```

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('train_data.csv')
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)
project_data.project_is_approved.value_counts()
```

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

Out[3]:

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

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

1.2 preprocessing of `project_subject_categories`

In [5]:

```
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 [6]:

```
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)

# 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]))
```

1.4 preprocessing of project grade categories

In [7]:

```
#preprocess project grade category
print(project_data['project_grade_category'].values[0])
print("="*50)
print(project_data['project_grade_category'].values[150])
print("="*50)
print(project_data['project_grade_category'].values[1000])
print("="*50)
print(project_data['project_grade_category'].values[20000])
print("="*50)


project_data['project_grade_category'].value_counts()
```

```
Grades PreK-2
==================================================
Grades 3-5
==================================================
Grades 3-5
==================================================
Grades PreK-2
==================================================
```

Out[7]:

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

In [8]:

```
preprocessed_project_grade_categories= []

for grade_cat in tqdm(project_data["project_grade_category"]):

    grade_cat = grade_cat.replace('-', '_')  #Replacing(-) with(_)
    grade_cat = grade_cat.replace('Grades', '') #Removing grades as it is redundant

    grad_cat = ' '.join(f for f in grade_cat.split())
    preprocessed_project_grade_categories.append(grad_cat.strip())
```

```
100%|██████████████████████████████| 109248/109248 [00:01<00:00, 86505.95it/s]
```

In [9]:

```
print(preprocessed_project_grade_categories[1])
print("="*50)
print(preprocessed_project_grade_categories[50])
print("="*50)
print(preprocessed_project_grade_categories[500])
print("="*50)
print(preprocessed_project_grade_categories[5000])
print("="*50)
print(preprocessed_project_grade_categories[10001])
print("="*50)
```

```
6_8
==================================================
PreK_2
==================================================
9_12
==================================================
PreK_2
==================================================
PreK_2
==================================================
```

1.5 preprocessing of teacher prefix

In [10]:

```
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna('null')
```

In [11]:

```
def replace_cate(lst):          # Removing (.) in Mrs.
    return lst.replace('.','')


project_data['teacher_prefix']= project_data['teacher_prefix'].astype(str).apply(replace_cate)
```

In [12]:

```
preprocessed_teacher_prefix = []

for teach_prefix in tqdm(project_data["teacher_prefix"]):

    preprocessed_teacher_prefix.append(teach_prefix.strip())
```

```
100%|██████████████████████████████| 109248/109248 [00:00<00:00, 297463.70it/s]
```

In [13]:

```
print(preprocessed_teacher_prefix[1])
print("="*50)
print(preprocessed_teacher_prefix[50])
print("="*50)
project_data.teacher_prefix.value_counts()
```

```
Mr
==================================================
Mrs
==================================================
```

Out[13]:

```
Mrs        57269
Ms         38955
Mr         10648
Teacher     2360
Dr            13
null           3
Name: teacher_prefix, dtype: int64
```

1.6 Adding a new feature Number of words in title

In [14]:

```
title_word_count = []
```

In [15]:

```
for a in project_data["project_title"] :
    b = len(a.split())
    title_word_count.append(b)
```

In [16]:

```
project_data["title_word_count"] = title_word_count
```

In [17]:

```
project_data.head(5)
```

Out[17]:

	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

combining 4 essays into 1 essay

In [18]:

```
# 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)
```

Adding a new feature Number of words in essay

In [19]:

```
essay_word_count=[]
```

In [20]:

```
for ess in project_data["essay"] :
    c = len(ess.split())
    essay_word_count.append(c)
```

In [21]:

```
project_data["essay_word_count"] = essay_word_count
```

In [22]:

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

Out[22]:

	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 [23]:

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

Out[23]:

	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

Train Test split

In [24]:

```
# train test split using sklearn.model_selection
from sklearn.model_selection import train_test_split

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'
],random_state=0)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train,
random_state=0)
```

In [25]:

```
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)
```

In [26]:

```
X_train.head(2)
```

Out[26]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_
75742	118221	p186156	f50f55a2b44b65b54f38f03c5df21922	Mrs	TX	2017-03-01 16:21:46	Gra
61001	57644	p180433	9e0fb5827f551d7e6966f8b3985e387b	Ms	NY	2017-03-09 10:19:06	G

1.8 Text preprocessing

In [27]:

```python
# printing some random reviews

print(X_train['essay'].values[0])
print("="*50)
print(X_train['essay'].values[500])
print("="*50)
```

```
My students are creative human beings.  They are sculpture and ceramic artists that have a passion
for 3-D art.  Many of my students are 2nd, 3rd, and 4th year sculpture students.  Some are even in
Advanced Placement and will receive college credit for their art.  Our school's minority
enrollment is 99% of the student body (majority Hispanic) and is a Title 1 school.  \r\nDespite th
e obstacles in their lives, my students use art as a means of expression, an educational tool, and
way to have their voices heard.   They live in the creative heart of San Antonio, next to the Blue
Star Arts Complex, and are being active agents for positive change in the community.It's no secret
that the arts are underfunded in schools.  In my sculpture class, we have limited access to techno
logy.  With a class set of i-pads, we could research artists and historical eras, photograph
artwork and create digital portfolios, and much more! Specifically for our spring installation, st
udents create stop-motion videos centered around a theme.  This year, students are tackling the su
bject of electronic waste.\r\n     By donating to our project, you will help students educate
their peers and community about the devastation caused by the unethical distribution and recycling
of electronic waste.  \r\nAlthough the theme changes each year, students can use i-pads for years
to come in order to raise social awareness through the arts.nannan
==================================================
Who are my students? Many of them have parents who attended this very same school, bringing their
children back to this rural community because of the trust they have in the teachers and
administration here. Still others come from families far away from here, with moms and/or dads tha
t have transferred to this area in search of work. I could say that many of them come from
financially comfortable homes, while just as many come from families who struggle financially. Man
y of them come from traditional families while others come from very nontraditional households. So
me of them have grown up insulated from violence and strife while others have been exposed to very
challenging situations. Lots of similarities… lots of differences.\n\nWhat really matters is no
matter who they are, when my students step into the music classroom, they step into a world of pos
sibilities. They are musicians in training... learning that they can do whatever they set their mi
nds to, that practice makes perfect and that commitment is vital for success. And they're learning
that connecting with people while making music is a joy that can be experienced through an entire
lifetime.This past summer, I was involved in a series of workshops which changed the way I teach m
usic. Having learned about Orff, Kodaly, Dalcroze-Eurhythmics and the Gordon approaches to
teaching music, it became clear that using any or all of these could have an incredible impact on
children in the music classroom.\r\n\r\nYour donation could provide us with a set of Orff
instruments which will involve and inspire even our youngest students in playing beautiful music.
We're starting small with one set and hope to build on this foundation in order to eventually
provide enough instruments for an entire classroom. Why Orff? They are exceptional, high quality,
durable instruments which will provide years of music instruction and student involvement in the c
lassroom. In addition, they have an incredibly beautiful tone.\r\n\r\nWith Orff instruments,
children in music class take turns playing simple pieces of music arranged especially for these
instruments. Sometimes, they simply come up with their own musical ideas to accompany a story or a
poem. Up to five students can play these Orff barred keyboard instruments while the rest of the cl
```

```
ass is engaged in playing non-pitched percussion. Together, they all sound like a \"percussion orc
hestra\" of both pitched and non-pitched percussion. Children learn to count beats, read/identify
& play notes accurately, listen to each other and stay together. They experience and learn about a
ll the elements of music while collaboratively creating beautiful music!\r\n\r\n\"Tell me, I forge
t,\r\nShow me, I remember.\r\nInvolve me, I understand\" --- Carl Orffnannan
==================================================
```

In [28]:

```python
# 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 [29]:

```python
sent = decontracted(X_train['essay'].values[20000])
print(sent)
print("="*50)
```

```
My school is located in the Appalachian Mountain region of VA. Over 50% of our students receive fr
ee or reduced price breakfast and lunch and may not eat dinner every evening.  Some are from secur
e families, who go to the movies together and get new shoes as their feet grow. Some of our childr
en view our school as their one safe place.\r\nEach morning I greet them at my classroom door with
a smile and they smile back ready to start a day of learning.  Thinking of their smiles gets me
out of bed and to school many mornings.\r\nThis year I plan to use Nursery Rhymes and other famili
ar stories to introduce concept of word skills to my students. I will write out the rhymes on sent
ence strips and put them in the pocket chart. \r\nThe children will be able to reach the sentences
, take them apart, put them back together and practice reading them. The Nursery Rhyme puzzles I h
ave requested fit perfectly with the way I plan to teach the rhymes. \r\nThe 100 pocket chart will
enable the students to practice counting with numbers and coins as they count their way to the 100
th day of school.nannan
==================================================
```

In [30]:

```python
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My school is located in the Appalachian Mountain region of VA. Over 50% of our students receive fr
ee or reduced price breakfast and lunch and may not eat dinner every evening.  Some are from secur
e families, who go to the movies together and get new shoes as their feet grow. Some of our childr
en view our school as their one safe place.  Each morning I greet them at my classroom door with a
smile and they smile back ready to start a day of learning.  Thinking of their smiles gets me out
of bed and to school many mornings.  This year I plan to use Nursery Rhymes and other familiar sto
ries to introduce concept of word skills to my students. I will write out the rhymes on sentence s
trips and put them in the pocket chart.   The children will be able to reach the sentences, take t
hem apart, put them back together and practice reading them. The Nursery Rhyme puzzles I have requ
ested fit perfectly with the way I plan to teach the rhymes.   The 100 pocket chart will enable th
e students to practice counting with numbers and coins as they count their way to the 100th day of
school.nannan
```

In [31]:

```python
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
```

```
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My school is located in the Appalachian Mountain region of VA Over 50 of our students receive free
or reduced price breakfast and lunch and may not eat dinner every evening Some are from secure fam
ilies who go to the movies together and get new shoes as their feet grow Some of our children view
our school as their one safe place Each morning I greet them at my classroom door with a smile and
they smile back ready to start a day of learning Thinking of their smiles gets me out of bed and t
o school many mornings This year I plan to use Nursery Rhymes and other familiar stories to introd
uce concept of word skills to my students I will write out the rhymes on sentence strips and put t
hem in the pocket chart The children will be able to reach the sentences take them apart put them
back together and practice reading them The Nursery Rhyme puzzles I have requested fit perfectly w
ith the way I plan to teach the rhymes The 100 pocket chart will enable the students to practice c
ounting with numbers and coins as they count their way to the 100th day of school nannan
```

In [32]:

```
# 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', "do
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"]
```

1.8.1 Preprocessesd training data - Text

In [33]:

```
# Combining all the above

from tqdm import tqdm
preprocessed_essays_train = []
# tqdm is for printing the status bar
for sentence in tqdm(X_train['essay'].values):
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    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

    preprocessed_essays_train.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████| 49041/49041 [02:20<00:00, 312.72it/s]
```

In [34]:

In [34]:

```
preprocessed_essays_test = []
# tqdm is for printing the status bar
for sentence in tqdm(X_test['essay'].values):
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    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

    preprocessed_essays_test.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████| 36052/36052 [01:38<00:00, 364.33it/s]
```

1.8.3 Preprocessed cross validation data

In [35]:

```
preprocessed_essays_cv = []
# tqdm is for printing the status bar
for sentence in tqdm(X_cv['essay'].values):
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    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

    preprocessed_essays_cv.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████| 24155/24155 [01:18<00:00, 309.51it/s]
```

1.9 preprocessing of project title

In [36]:

```
# printing some randomproject titles.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[20000])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
```

In [37]:

```
title = decontracted(X_train['project_title'].values[2000])
```

1.9.1 Preprocessing of Project Title(Train)

In [38]:

```
preprocessed_titles_train = []

for titles in tqdm(X_train["project_title"]):
    title = ' '.join(f for f in title.split() if f not in stopwords)
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)

    preprocessed_titles_train.append(title.lower().strip())
```

```
100%|██████████████████████████████████████| 49041/49041 [00:07<00:00, 6601.05it/s]
```

1.9.2 Preprocessing of Project Title(Test)

In [39]:

```
preprocessed_titles_test = []

for titles in tqdm(X_test["project_title"]):
    title = ' '.join(f for f in title.split() if f not in stopwords)
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)

    preprocessed_titles_test.append(title.lower().strip())
```

```
100%|██████████████████████████████████████| 36052/36052 [00:05<00:00, 6703.84it/s]
```

In [40]:

```
preprocessed_titles_test[10]
```

Out[40]:

```
'bearcat book club'
```

1.9.2 Preprocessing of Project Title(CV)

In [41]:

```
preprocessed_titles_cv = []

for titles in tqdm(X_cv["project_title"]):
    title = ' '.join(f for f in title.split() if f not in stopwords)
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)

    preprocessed_titles_cv.append(title.lower().strip())
```

```
100%|██████████████████████████████████████| 24155/24155 [00:03<00:00, 6040.51it/s]
```

In [42]:

```
preprocessed_titles_cv[600]
```

Out[42]:

[illegible]

```
'yum raspberry pi for all'
```

1.5 Preparing data for models

In [43]:

```
project_data.columns
```

Out[43]:

```
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', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'title_word_count', 'essay',
       'essay_word_count'],
      dtype='object')
```

we are going to consider

```
      - school_state : categorical data
      - clean_categories : categorical data
      - clean_subcategories : categorical data
      - project_grade_category : categorical data
      - teacher_prefix : categorical data

      - project_title : text data
      - text : text data
      - project_resource_summary: text data (optinal)

      - quantity : numerical (optinal)
      - teacher_number_of_previously_posted_projects : numerical
      - price : numerical
```

1.5.1 Vectorizing Categorical data

one hot vector for clean categories of Projects (train,test,cv)

In [44]:

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

from sklearn.feature_extraction.text import CountVectorizer

vectorizer_proj = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary
=True)
vectorizer_proj.fit(X_train['clean_categories'].values)

categories_one_hot_train = vectorizer_proj.transform(X_train['clean_categories'].values)
categories_one_hot_test = vectorizer_proj.transform(X_test['clean_categories'].values)
categories_one_hot_cv = vectorizer_proj.transform(X_cv['clean_categories'].values)

print(vectorizer_proj.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",categories_one_hot_test.shape)
print("Shape of matrix of CV data after one hot encoding ",categories_one_hot_cv.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix of Train data after one hot encoding  (49041, 9)
Shape of matrix of Test data after one hot encoding  (36052, 9)
Shape of matrix of CV data after one hot encoding  (24155, 9)
```

one hot vector for clean subcategories (train ,test,cv)

In [45]:

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

vectorizer_sub_proj = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False
, binary=True)
vectorizer_sub_proj.fit(X_train['clean_subcategories'].values)

sub_categories_one_hot_train = vectorizer_sub_proj.transform(X_train['clean_subcategories'].values
)
sub_categories_one_hot_test = vectorizer_sub_proj.transform(X_test['clean_subcategories'].values)
sub_categories_one_hot_cv = vectorizer_sub_proj.transform(X_cv['clean_subcategories'].values)


print(vectorizer_sub_proj.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",sub_categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",sub_categories_one_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",sub_categories_one_hot_cv
.shape)
```

```
['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 of Train data after one hot encoding  (49041, 30)
Shape of matrix of Test data after one hot encoding  (36052, 30)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 30)
```

One hot vector for school states(train,test,cv)

In [46]:

```
my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())
```

In [47]:

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

In [48]:

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

vectorizer_states = CountVectorizer(vocabulary=list(sorted_school_state_cat_dict.keys()),
lowercase=False, binary=True)
vectorizer_states.fit(X_train['school_state'].values)

school_state_categories_one_hot_train = vectorizer_states.transform(X_train['school_state'].values
)
school_state_categories_one_hot_test = vectorizer_states.transform(X_test['school_state'].values)
school_state_categories_one_hot_cv = vectorizer_states.transform(X_cv['school_state'].values)

print(vectorizer_states.get_feature_names())

print("Shape of matrix of Train data after one hot encoding
",school_state_categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",school_state_categories_one_hot_test.
shape)
print("Shape of matrix of Cross Validation data after one hot encoding
",school_state_categories_one_hot_cv.shape)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'I
A', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ',
'NJ', 'OK', 'WA', 'MA', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX
', 'CA']
Shape of matrix of Train data after one hot encoding  (49041, 51)
Shape of matrix of Test data after one hot encoding  (36052, 51)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 51)
```

one hot vector for Project grade category (train,test,cv)

In [49]:

```
my_counter = Counter()
for project_grade in preprocessed_project_grade_categories:
    my_counter.update(project_grade.split())
```

In [50]:

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

In [51]:

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

vectorizer_grade = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()),
lowercase=False, binary=True)
vectorizer_grade.fit(X_train['project_grade_category'].values)

project_grade_categories_one_hot_train =
vectorizer_grade.transform(X_train['project_grade_category'].values)
project_grade_categories_one_hot_test = vectorizer_grade.transform(X_test['project_grade_category'
].values)
project_grade_categories_one_hot_cv = vectorizer_grade.transform(X_cv['project_grade_category'].va
lues)

print(vectorizer_grade.get_feature_names())

print("Shape of matrix of Train data after one hot encoding
",project_grade_categories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",project_grade_categories_one_hot_test
.shape)
print("Shape of matrix of Cross Validation data after one hot encoding
",project_grade_categories_one_hot_cv.shape)
```

```
['9_12', '6_8', '3_5', 'PreK_2']
Shape of matrix of Train data after one hot encoding  (49041, 4)
Shape of matrix of Test data after one hot encoding  (36052, 4)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 4)
```

One hot vector for teacher prefix(train,test,cv)

In [52]:

```
vectorizer_teacher = CountVectorizer()
vectorizer_teacher.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
teacher_prefix_categories_one_hot_train = vectorizer_teacher.transform(X_train['teacher_prefix'].v
alues)
teacher_prefix_categories_one_hot_cv = vectorizer_teacher.transform(X_cv['teacher_prefix'].values)
teacher_prefix_categories_one_hot_test =
vectorizer_teacher.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print("Shape of matrix of Train data after one hot
encoding",teacher_prefix_categories_one_hot_train.shape, y_train.shape)
print("Shape of matrix of cv data after one hot encoding",teacher_prefix_categories_one_hot_cv.sha
pe, y_cv.shape)
print("Shape of matrix of Test data after one hot encoding",teacher_prefix_categories_one_hot_test
print("Shape of matrix of Test data after one hot encoding",teacher_prefix_categories_one_hot_test
.shape, y_test.shape)
print(vectorizer_teacher.get_feature_names())
print("="*100)
```

```
After vectorizations
Shape of matrix of Train data after one hot encoding (49041, 6) (49041,)
Shape of matrix of cv data after one hot encoding (24155, 6) (24155,)
Shape of matrix of Test data after one hot encoding (36052, 6) (36052,)
['dr', 'mr', 'mrs', 'ms', 'null', 'teacher']
===================================================================================================
```

1.11 Vectorizing text data

A) Bag of words

BOW train data essays

In [53]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_essay = CountVectorizer(min_df=10,max_features=5000)  #selecting top 5000 features
vectorizer_bow_essay.fit(preprocessed_essays_train)

text_bow_train = vectorizer_bow_essay.transform(preprocessed_essays_train)

print("Shape of matrix after one hot encoding ",text_bow_train.shape)
```

```
Shape of matrix after one hot encoding  (49041, 5000)
```

BOW (test essays)

In [54]:

```
text_bow_test = vectorizer_bow_essay.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 5000)
```

Bow (cv essays)

In [55]:

```
text_bow_cv = vectorizer_bow_essay.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encoding ",text_bow_cv.shape)
```

```
Shape of matrix after one hot encoding  (24155, 5000)
```

Bow(train titles)

In [56]:

```
vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit(preprocessed_titles_train)
title_bow_train = vectorizer_bow_title.transform(preprocessed_titles_train)
print("Shape of matrix after one hot encoding ",title_bow_train.shape)
```

```
Shape of matrix after one hot encoding  (49041, 2105)
```

Bow(test titles)

In [57]:

```
title_bow_test = vectorizer_bow_title.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encoding ",title_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 2105)
```

Bow(cv titles)

In [58]:

```
title_bow_cv = vectorizer_bow_title.transform(preprocessed_titles_cv)
print("Shape of matrix after one hot encoding ",title_bow_cv.shape)
```

```
Shape of matrix after one hot encoding  (24155, 2105)
```

B) Tfidf

tfidf(train essays)

In [59]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_essay = TfidfVectorizer(min_df=10,max_features=5000) #Considering top 5000
features
vectorizer_tfidf_essay.fit(preprocessed_essays_train)

text_tfidf_train = vectorizer_tfidf_essay.transform(preprocessed_essays_train)
print("Shape of matrix after one hot encoding ",text_tfidf_train.shape)
```

```
Shape of matrix after one hot encoding  (49041, 5000)
```

tfidf(test essays)

In [60]:

```
text_tfidf_test = vectorizer_tfidf_essay.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 5000)
```

tfidf(cv essays)

In [61]:

```
text_tfidf_cv = vectorizer_tfidf_essay.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encoding ",text_tfidf_cv.shape)
```

```
Shape of matrix after one hot encoding  (24155, 5000)
```

tfidf(train titles)

In [62]:

```
vectorizer_tfidf_titles = TfidfVectorizer(min_df=10)

vectorizer_tfidf_titles.fit(preprocessed_titles_train)
vectorizer_tfidf_titles.fit(preprocessed_titles_train)
title_tfidf_train = vectorizer_tfidf_titles.transform(preprocessed_titles_train)
print("Shape of matrix after one hot encoding ",title_tfidf_train.shape)
```

```
Shape of matrix after one hot encoding  (49041, 2105)
```

tfidf(test titles)

In [63]:

```
title_tfidf_test = vectorizer_tfidf_titles.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encoding ",title_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 2105)
```

tfidf(cv titles)

In [64]:

```
title_tfidf_cv = vectorizer_tfidf_titles.transform(preprocessed_titles_cv)
print("Shape of matrix after one hot encoding ",title_tfidf_cv.shape)
```

```
Shape of matrix after one hot encoding  (24155, 2105)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [ ]:

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

train essays

In [ ]:

```
# average Word2Vec
# compute average word2vec for each review.

avg_w2v_vectors_train = [];

for sentence in tqdm(preprocessed_essays_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))
```

test essays

In [ ]:

```
# average Word2Vec
# compute average word2vec for each review.

avg_w2v_vectors_test = [];

for sentence in tqdm(preprocessed_essays_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))
```

cv essays

In [ ]:

```
# average Word2Vec
# compute average word2vec for each review.

avg_w2v_vectors_cv = [];

for sentence in tqdm(preprocessed_essays_cv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)

print(len(avg_w2v_vectors_cv))
print(len(avg_w2v_vectors_cv[0]))
```

train titles

In [ ]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_train): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_train.append(vector)

print(len(avg_w2v_vectors_titles_train))
print(len(avg_w2v_vectors_titles_train[0]))
```

test titles

In [ ]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_test = []; # the avg-w2v for each sentence/review is stored in this list
```

```
avg_w2v_vectors_titles_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_test.append(vector)

print(len(avg_w2v_vectors_titles_test))
print(len(avg_w2v_vectors_titles_test[0]))
```

Cv titles

In [ ]:

```
# Similarly you can vectorize for title also

avg_w2v_vectors_titles_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_cv): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_cv.append(vector)

print(len(avg_w2v_vectors_titles_cv))
print(len(avg_w2v_vectors_titles_cv[0]))
```

using pretrained models : Tfidf weighted W2V

train essays

In [ ]:

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

In [ ]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_train.append(vector)
```

```
print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))
```

test essays

In [ ]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))
```

cv essays

In [ ]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_cv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_cv.append(vector)

print(len(tfidf_w2v_vectors_cv))
print(len(tfidf_w2v_vectors_cv[0]))
```

train titles

In [ ]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles_train)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [ ]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_titles_train = [];

for sentence in tqdm(preprocessed_titles_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_train.append(vector)

print(len(tfidf_w2v_vectors_titles_train))
```

test titles

In [ ]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_titles_test = [];

for sentence in tqdm(preprocessed_titles_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_test.append(vector)

print(len(tfidf_w2v_vectors_titles_test))
print(len(tfidf_w2v_vectors_titles_test[0]))
```

cv titles

In [ ]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_titles_cv = [];

for sentence in tqdm(preprocessed_titles_cv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_cv.append(vector)
```

```
print(len(tfidf_w2v_vectors_titles_cv))
print(len(tfidf_w2v_vectors_titles_cv[0]))
```

1.12 Vectorizing Numerical features

Various numerical feautures are :

1.Price

2.Quantity

3.Number of Projects previously proposed by Teacher

4.Title word Count (introduced by us)

5.Essay word Count (introduced by us)

1) Price

In [65]:

```
# 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(4)
```

Out[65]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21
2	p000003	298.97	4
3	p000004	1113.69	98

In [66]:

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

In [67]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

normalizer.fit(X_train['price'].values.reshape(1,-1))

price_train = normalizer.transform(X_train['price'].values.reshape(1,-1))
price_cv = normalizer.transform(X_cv['price'].values.reshape(1,-1))
price_test = normalizer.transform(X_test['price'].values.reshape(1,-1))

print("After vectorizations")
print(price_train.shape, y_train.shape)
print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 49041) (49041,)
(1, 24155) (24155,)
(1, 36052) (36052,)
====================================================================================================
```

2) Quantity

In [68]:

```
normalizer = Normalizer()

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

normalizer.fit(X_train['quantity'].values.reshape(1,-1))

quantity_train = normalizer.transform(X_train['quantity'].values.reshape(1,-1))
quantity_cv = normalizer.transform(X_cv['quantity'].values.reshape(1,-1))
quantity_test = normalizer.transform(X_test['quantity'].values.reshape(1,-1))

print("After vectorizations")
print(quantity_train.shape, y_train.shape)
print(quantity_cv.shape, y_cv.shape)
print(quantity_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 49041) (49041,)
(1, 24155) (24155,)
(1, 36052) (36052,)
====================================================================================================
```

3) Number of Projects previously proposed by Teacher

In [69]:

```
normalizer = Normalizer()

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

normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

prev_projects_train = normalizer.transform(X_train['teacher_number_of_previously_posted_projects']
.values.reshape(-1,1))
prev_projects_cv =
normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_projects_test = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].v
alues.reshape(-1,1))

print("After vectorizations")
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
```

```
(36052, 1) (36052,)
====================================================================================================
```

4) title word count

In [70]:

```
normalizer = Normalizer()

normalizer.fit(X_train['title_word_count'].values.reshape(1,-1))

title_word_count_train = normalizer.transform(X_train['title_word_count'].values.reshape(1,-1))
title_word_count_cv = normalizer.transform(X_cv['title_word_count'].values.reshape(1,-1))
title_word_count_test = normalizer.transform(X_test['title_word_count'].values.reshape(1,-1))

print("After vectorizations")
print(title_word_count_train.shape, y_train.shape)
print(title_word_count_cv.shape, y_cv.shape)
print(title_word_count_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 49041) (49041,)
(1, 24155) (24155,)
(1, 36052) (36052,)
====================================================================================================
```

5) essay word count

In [71]:

```
normalizer = Normalizer()

normalizer.fit(X_train['essay_word_count'].values.reshape(1,-1))

essay_word_count_train = normalizer.transform(X_train['essay_word_count'].values.reshape(1,-1))
essay_word_count_cv = normalizer.transform(X_cv['essay_word_count'].values.reshape(1,-1))
essay_word_count_test = normalizer.transform(X_test['essay_word_count'].values.reshape(1,-1))

print("After vectorizations")
print(essay_word_count_train.shape, y_train.shape)
print(essay_word_count_cv.shape, y_cv.shape)
print(essay_word_count_test.shape, y_test.shape)
```

```
After vectorizations
(1, 49041) (49041,)
(1, 24155) (24155,)
(1, 36052) (36052,)
```

Assignment 4

1. Apply Multinomial NaiveBayes on these feature sets
    - : categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - : categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

1. 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

1. Feature importance
    - : Find the top 10 features of positive class and top 10 features of negative class for both feature sets and using values of `feature_log_prob_` parameter of MultinomialNB and print their corresponding feature names

1. 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.

1. 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

2. Naive Bayes

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

In [76]:

```
# reshaping to appt shape so that it can be used with hstack
```

In [77]:

```
price_train = (X_train['price'].values.reshape(-1,1))
price_cv = (X_cv['price'].values.reshape(-1,1))
price_test = (X_test['price'].values.reshape(-1,1))

quantity_train =(X_train['quantity'].values.reshape(-1,1))
quantity_cv = (X_cv['quantity'].values.reshape(-1,1))
quantity_test = (X_test['quantity'].values.reshape(-1,1))

prev_projects_train = (X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)
)
prev_projects_cv = (X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_projects_test = (X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

title_word_count_train = (X_train['title_word_count'].values.reshape(-1,1))
title_word_count_cv = (X_cv['title_word_count'].values.reshape(-1,1))
title_word_count_test = (X_test['title_word_count'].values.reshape(-1,1))


essay_word_count_train = (X_train['essay_word_count'].values.reshape(-1,1))
essay_word_count_cv = (X_cv['essay_word_count'].values.reshape(-1,1))
essay_word_count_test = (X_test['essay_word_count'].values.reshape(-1,1))
```

In [78]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train, project_grade_categories_one_hot_train,
teacher_prefix_categories_one_hot_train, price_train, quantity_train, prev_projects_train, title_wo
rd_count_train, essay_word_count_train, title_bow_train, text_bow_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test, project_grade_categories_one_hot_test,
teacher_prefix_categories_one_hot_test, price_test, quantity_test, prev_projects_test,
title_word_count_test, essay_word_count_test, title_bow_test, text_bow_test)).tocsr()
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv, project_grade_categories_one_hot_cv,
teacher_prefix_categories_one_hot_cv, price_cv, quantity_cv, prev_projects_cv, title_word_count_cv,
essay_word_count_cv, title_bow_cv, text_bow_cv)).tocsr()
```

In [79]:

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

```
Final Data matrix
(49041, 7210) (49041,)
(24155, 7210) (24155,)
(36052, 7210) (36052,)
====================================================================================================
```

In [80]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

A) Random alpha values (hyperparameter)

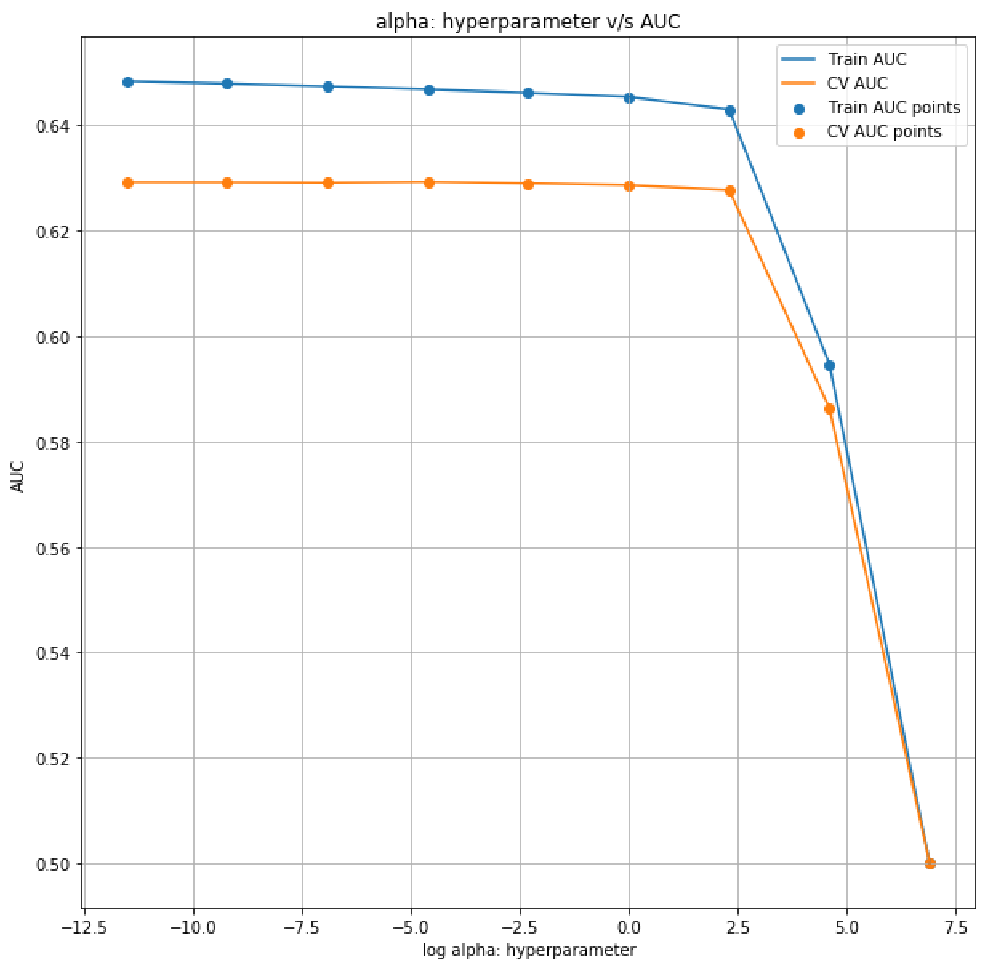
In [81]:

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

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

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

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

    y_train_pred = batch_predict(nb, X_tr)
    y_cv_pred = batch_predict(nb, X_cr)


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability 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 = math.log(a)
    log_alphas.append(b)
```

```
100%|████████████████████████████████████████████| 9/9 [00:10<00:00,  1.12s/it]
100%|██████████████████████████████████████████| 9/9 [00:00<00:00, 4495.50it/s]
```

In [82]:

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

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

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



summary

- We have started with hyperparameter alpha with as low as 0.0001 to 1000.Since it is difficult to plot the given range we have used log alphas on x-axis and Auc on y axis as shown in the plot.
- One of the main reason for using log scale is log scales allow a large range to be displayed without small values being compressed down into bottom of the graph.
- we observe that as log alpha approaches close to 7 ,both train AUc and cv AUC lines converge
- Using this plot we see after alpha=10 both lines converge at amuch higher rate

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

- return_train_score=True needs to be explicitly set for GridSearchCV function to return train scores
- Also verbose is set to 2 to display progress messages

In [83]:

```
from sklearn.model_selection import GridSearchCV

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

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

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

clf.fit(X_tr, y_train)

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

```
Fitting 10 folds for each of 11 candidates, totalling 110 fits
```

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

```
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.6s
```

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

```
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.5s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.5s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.5s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.6s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.5s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.6s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.6s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.4s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.4s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.3s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.4s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.4s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.4s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.3s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.4s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.4s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.3s
[CV] alpha=0.001 .....................................................
[CV]                                        alpha=0.001, total=   0.5s
```

```
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.4s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.4s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.3s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.3s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.3s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.3s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.3s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.3s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.4s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.4s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.3s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.3s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.3s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.3s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.3s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.3s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.3s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.2s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.5s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.5s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.4s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.4s
[CV] alpha=0.8
```

```
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.5s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.5s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.4s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.5s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.4s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.5s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.4s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.4s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.4s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.4s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.4s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.5s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.5s
[CV] alpha=1 .........................................................
[CV] .......................................... alpha=1, total=   0.4s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.4s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.4s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.4s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.5s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.5s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.5s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.5s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.5s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.5s
[CV] alpha=10 ........................................................
[CV] ......................................... alpha=10, total=   0.4s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.4s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.4s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.4s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.4s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.4s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.5s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.5s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.5s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.4s
[CV] alpha=100 .......................................................
[CV] ........................................ alpha=100, total=   0.5s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.4s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.4s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.5s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.5s
[CV] alpha=1000 ......................................................
[CV] ....................................... alpha=1000, total=   0.5s
```

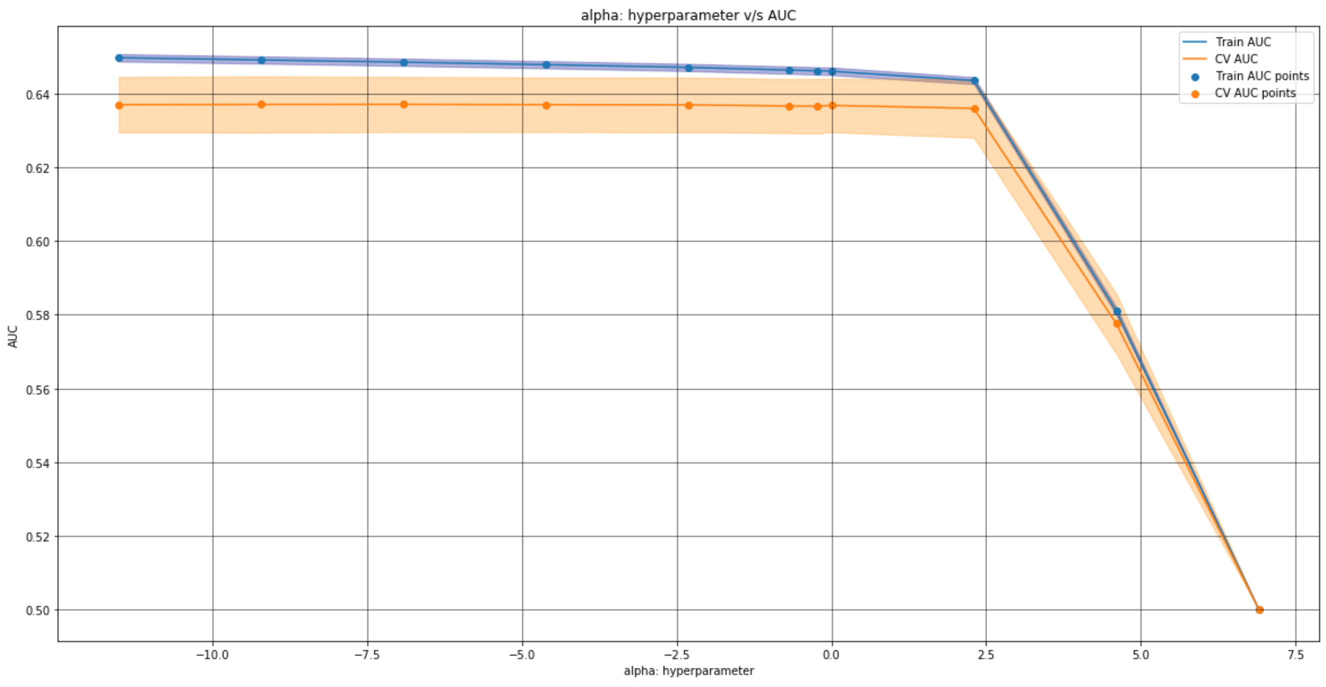
```
[CV] ....................................... alpha=1000, total=   0.5s
[CV] alpha=1000 .......................................................
[CV] ....................................... alpha=1000, total=   0.5s
[CV] alpha=1000 .......................................................
[CV] ....................................... alpha=1000, total=   0.5s
[CV] alpha=1000 .......................................................
[CV] ....................................... alpha=1000, total=   0.4s
[CV] alpha=1000 .......................................................
[CV] ....................................... alpha=1000, total=   0.5s
[CV] alpha=1000 .......................................................
[CV] ....................................... alpha=1000, total=   0.5s
```

```
[Parallel(n_jobs=1)]: Done 110 out of 110 | elapsed:  1.2min finished
```

In [84]:

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

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

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

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

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

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


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

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



summary

- We have started with hyperparameter alpha with as low as 0.0001 to 1000.Since it is difficult to plot the given range we have used log alphas on x-axis and Auc on y axis as shown in the plot.
- One of the main reason for using log scale is log scales allow a large range to be displayed without small values being compressed down into bottom of the graph.
- we observe that as log alpha approaches close to 7 ,both train AUc and cv AUC lines converge
- Using this plot we see after alpha=100 both lines converge at amuch higher rate

Train model using the best hyper-parameter value

- Using best*params* attribute of gridsearch cv we can obtain the optimal value of alpha among the values we have selected
- It simplifes our task and we can be rest assured that selected hyperparameter is most optimal one

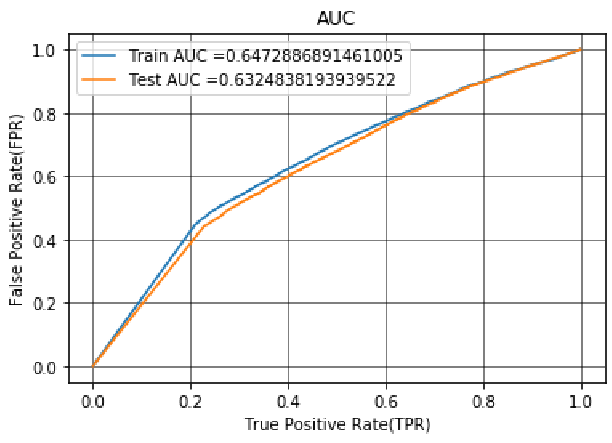
In [85]:

```
best_alpha1=0.001
```

```
{'alpha': 0.001}
```

In [91]:

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

nb_bow = MultinomialNB(alpha = 0.001,class_prior=[0.5,0.5])

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

y_train_pred = batch_predict(nb_bow, X_tr)
y_test_pred = batch_predict(nb_bow, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



summary

- For Bow model for alpha=0.001 ,we get train AUC of 0.64 and Test AUC of 0.63

D) confusion matrix

In [92]:

```
def predict(proba, threshould, fpr, tpr):

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

    # (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
```

train data

In [93]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.922
[[ 3713  3713]
 [12350 29265]]
```

In [94]:

```
conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds,
train_fpr, train_fpr)), range(2),range(2))
```

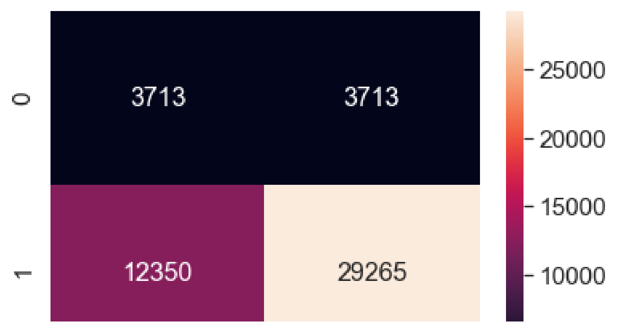
```
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.922
```

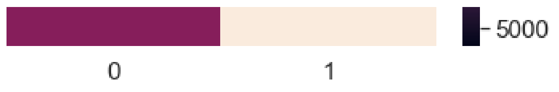
In [95]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[95]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x6168012cf8>
```





Summary on train data

- In the following confusion matrix we observe that the model has 29265 true positives while true negatives are only 3713
  - It has large number of false negatives which are roughly close to 12000

test data

In [96]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 1.0
[[ 3081  2378]
 [11297 19296]]
```

In [97]:

```
conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, tes
t_fpr, test_fpr)), range(2),range(2))
```

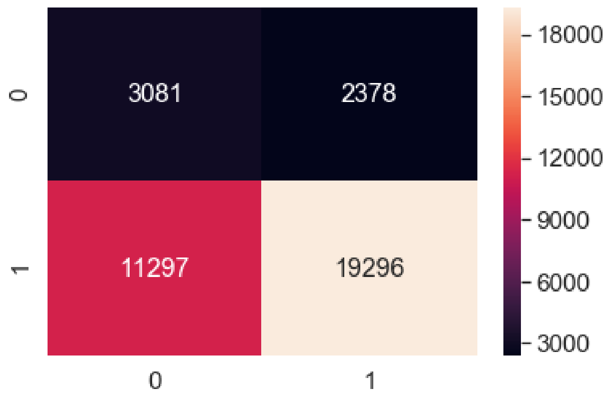
```
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 1.0
```

In [98]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[98]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x610c991828>
```



Summary on test data

- The number of true positives dominate ,thera are 19296 in number,
- The least number among 4 quantites is false positives which are 3100 false positives
- similar trend is observed for false negatives which are roughly 11000

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

In [99]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train, project_grade_categories_one_hot_train,
teacher_prefix_categories_one_hot_train, price_train, quantity_train, prev_projects_train, title_wo
rd_count_train, essay_word_count_train, text_tfidf_train, title_tfidf_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test, project_grade_categories_one_hot_test,
teacher_prefix_categories_one_hot_test, price_test, quantity_test, prev_projects_test,
title_word_count_test, essay_word_count_test, text_tfidf_test, title_tfidf_test)).tocsr()
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv, project_grade_categories_one_hot_cv,
teacher_prefix_categories_one_hot_cv, price_cv, quantity_cv, prev_projects_cv, title_word_count_cv,
essay_word_count_cv, text_tfidf_cv, title_tfidf_cv)).tocsr()
```

In [100]:

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

```
Final Data matrix
(49041, 7210) (49041,)
(24155, 7210) (24155,)
(36052, 7210) (36052,)
====================================================================================================
```

A) random alpha values

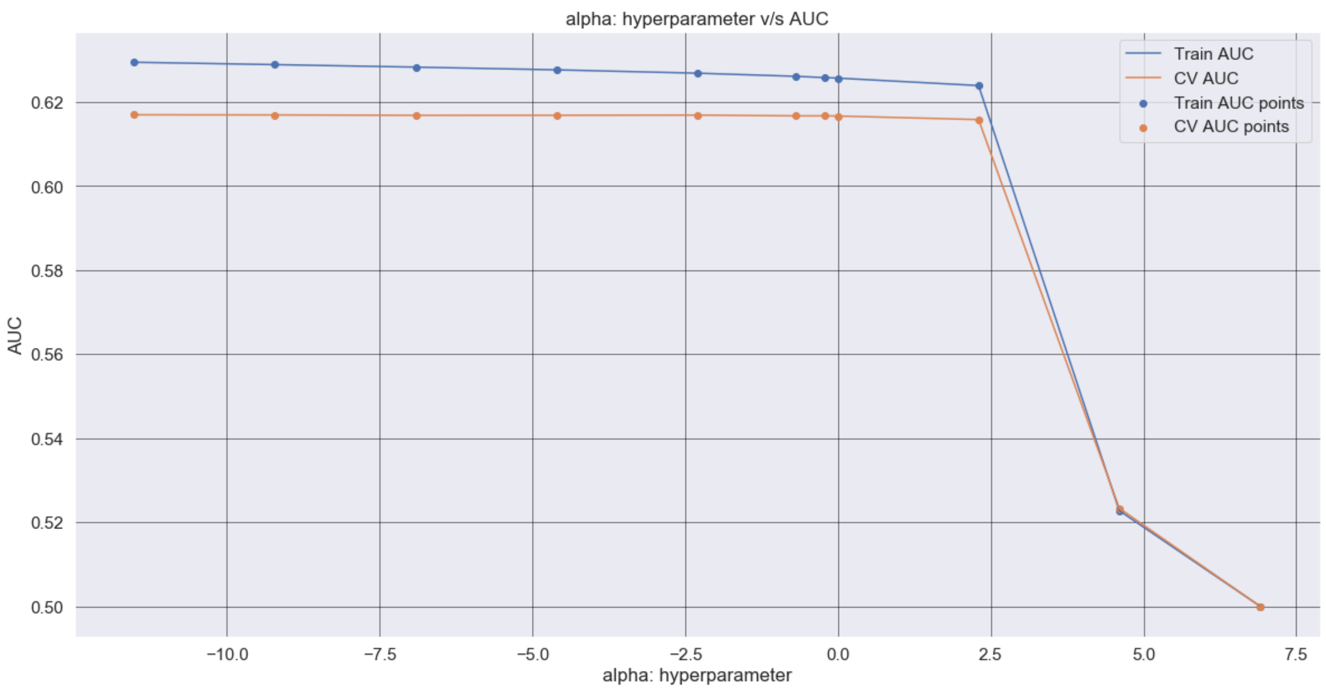
In [101]:

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

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

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

    y_train_pred = batch_predict(nb, X_tr)
    y_cv_pred = batch_predict(nb, X_cr)


    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability 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 = math.log(a)
    log_alphas.append(b)
```

```
100%|██████████████████████████████████████████| 11/11 [00:11<00:00,  1.05s/it]
100%|████████████████████████████████████████| 11/11 [00:00<00:00, 11003.42it/s]
```

In [102]:

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



summary

- We have started with hyperparameter alpha with as low as 0.0001 to 1000.Since it is difficult to plot the given range we have used log alphas on x-axis and Auc on y axis as shown in the plot.
- One of the main reason for using log scale is log scales allow a large range to be displayed without small values being compressed down into bottom of the graph.
- we observe that as log alpha approaches close to 7 ,both train AUc and cv AUC lines converge
- Using this plot we see after alpha=100 both lines converge at a much higher rate
- One thing different from set1 plot is Cv Auc line remains constant for a long time ,only after alpha=10 it starts dropping

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

In [103]:

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

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

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

clf.fit(X_tr, y_train)

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

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
Fitting 10 folds for each of 10 candidates, totalling 100 fits
```

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

```
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.5s
```

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

```
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.5s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.5s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.6s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.5s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.4s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.3s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.4s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.4s
[CV] alpha=1e-05 .....................................................
[CV] ...................................... alpha=1e-05, total=   0.3s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.3s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.3s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.3s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.2s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.3s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.0001 ....................................................
[CV] ..................................... alpha=0.0001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.001 .....................................................
[CV] ...................................... alpha=0.001, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
```

```
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.5s
[CV] alpha=0.01 ......................................................
[CV] ....................................... alpha=0.01, total=   0.4s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.4s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.3s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.1 .......................................................
[CV] ........................................ alpha=0.1, total=   0.5s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.5s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.4s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.4s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.3s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.3s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.3s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.3s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.25 ......................................................
[CV] ....................................... alpha=0.25, total=   0.2s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.5s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.5s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.4s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.5s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.4s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.5s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.6s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.6s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.5s
[CV] alpha=0.5 .......................................................
[CV] ........................................ alpha=0.5, total=   0.5s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.6s
[CV] alpha=0.8 .......................................................
[CV] ........................................ alpha=0.8, total=   0.5s
[CV] alpha=0.8 .......................................................
```

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

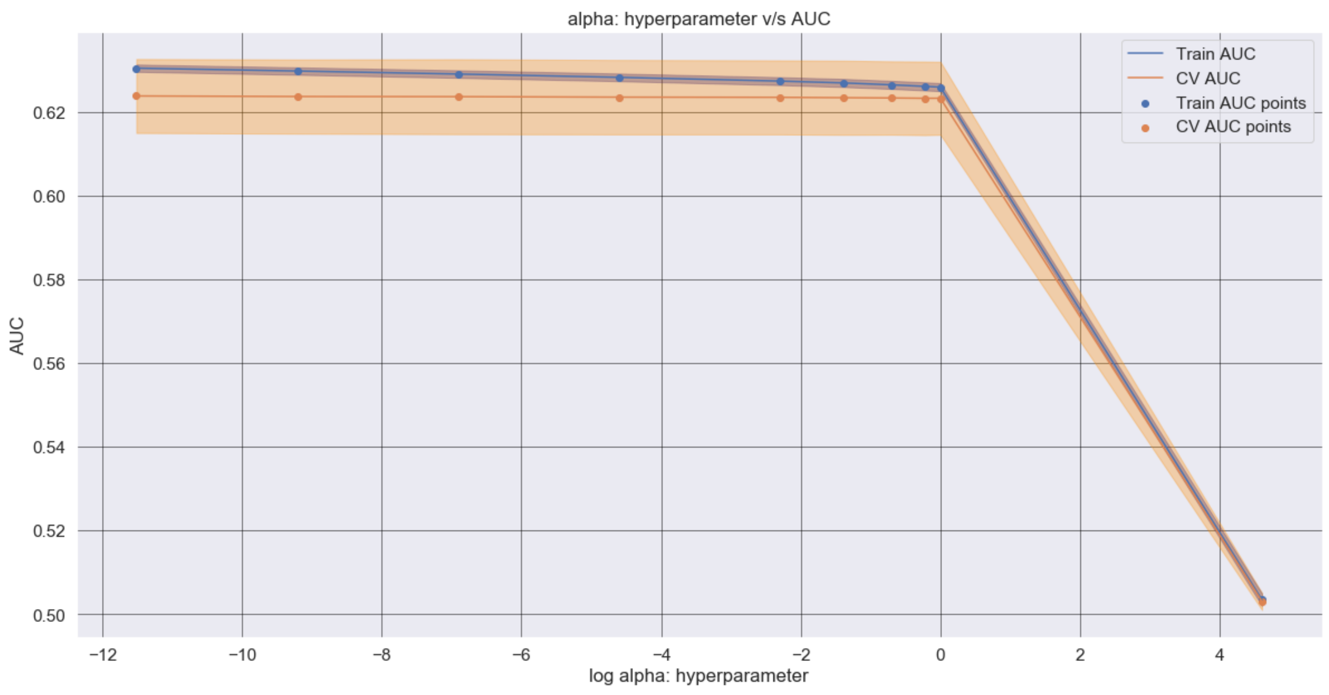
```
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:  1.1min finished
```

In [104]:

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

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

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

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

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

```
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alphas,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkoran
ge')

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


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

```
100%|██████████████████████████████████████████| 10/10 [00:00<00:00, 9984.06it/s]
```



summary

- We have started with hyperparameter alpha with as low as 0.0001 to 1000.Since it is difficult to plot the given range we have used log alphas on x-axis and Auc on y axis as shown in the plot.
- One of the main reason for using log scale is log scales allow a large range to be displayed without small values being compressed down into bottom of the graph.
- we observe that as log alpha approaches close to 2 ,both train AUc and cv AUC lines converge
- Using this plot we see after alpha=100 both lines converge at amuch higher rate
- One thing different from set1 plot is Cv Auc line remains constant for a long time ,only after alpha=0.1 it starts dropping

C) Train model using the best hyper-parameter value of alpha

In [ ]:

```
# Code idea for best_params_ taken from here
https://datascience.stackexchange.com/questions/21877/how-to-use-the-output-of-gridsearch
```

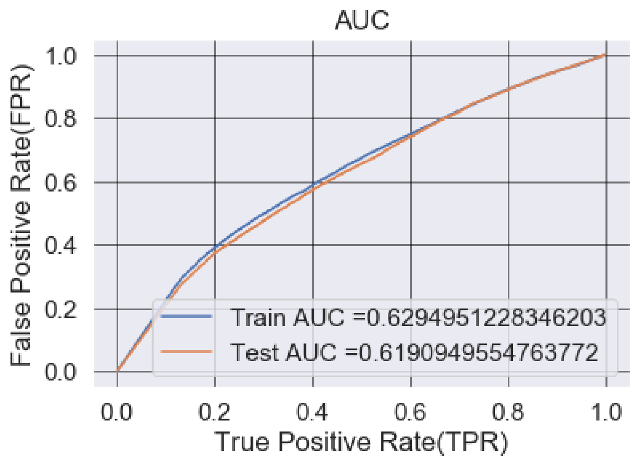
In [105]:

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

```
{'alpha': 1e-05}
```

In [109]:

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

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

y_train_pred = batch_predict(nb_tfidf, X_tr)
y_test_pred = batch_predict(nb_tfidf, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



Summary

- From given plot we observe that at alpha=1e-05 we get train AUC of 0.62 and test AUC of 0.61

D) Confusion matrix

train data

In [110]:

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.96
[[ 3713  3713]
 [13647 27968]]
```

In [111]:

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

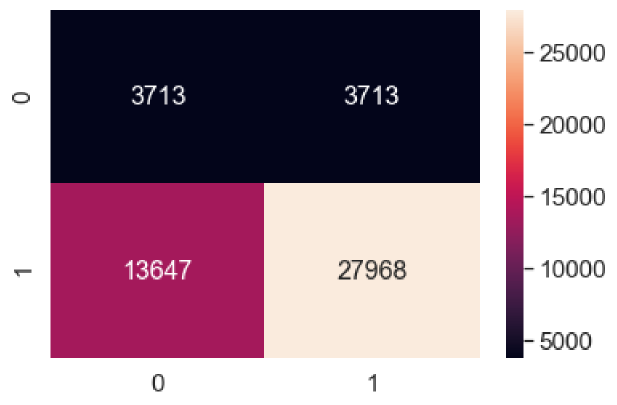
```
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.96
```

In [112]:

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

Out[112]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x611a95f400>
```



summary

- For training data we get roughly 28000 true positives and 3713 true negatives and false positives each
- Again we have roughly 14000 false negatives which are alot in number

test data

In [113]:

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
====================================================================================================

Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 1.0
[[ 3244  2215]
 [12931 17662]]
```

In [114]:

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

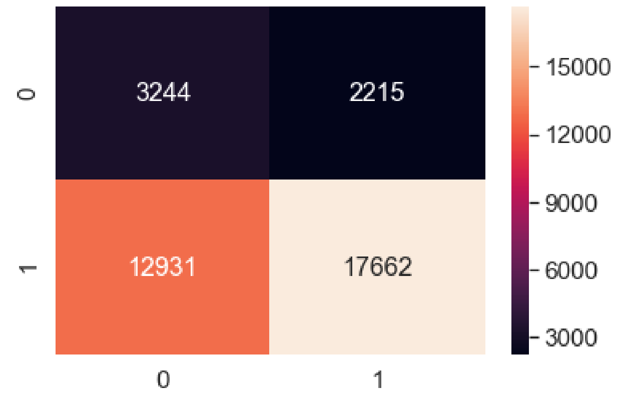
```
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 1.0
```

In [115]:

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

Out[115]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x610f111710>
```



summary

- we have roughly 17662 true positives for test data and roughly 2200 true negatives
- Again false negatives are pretty high in number(13k)

Select best 30 features of both Positive and negative class for both the sets of data

set1 Bow

In [116]:

```
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train, project_grade_categories_one_hot_train,
teacher_prefix_categories_one_hot_train, price_train, quantity_train, prev_projects_train, title_wo
rd_count_train, essay_word_count_train, title_bow_train, text_bow_train)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test, project_grade_categories_one_hot_test,
teacher_prefix_categories_one_hot_test, price_test, quantity_test, prev_projects_test,
title_word_count_test, essay_word_count_test, title_bow_test, text_bow_test)).tocsr()
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv, project_grade_categories_one_hot_cv,
teacher_prefix_categories_one_hot_cv, price_cv, quantity_cv, prev_projects_cv, title_word_count_cv,
essay_word_count_cv, title_bow_cv, text_bow_cv)).tocsr()
```

In [117]:

```
nb_bow = MultinomialNB(alpha = 0.001,class_prior=[0.5,0.5])

nb_bow.fit(X_tr, y_train)
```

Out[117]:

```
MultinomialNB(alpha=0.001, class_prior=[0.5, 0.5], fit_prior=True)
```

In [118]:

```
# Collecting feature names for BOW set1
# adding to end of list by concateing features
# Code snippet taken from here https://stackabuse.com/append-vs-extend-in-python-lists/

bow_features_names1 = []
```

In [119]:

```
for cnt in vectorizer_proj.get_feature_names() :
    bow_features_names1.append(cnt)

for cnt1 in vectorizer_sub_proj.get_feature_names() :
    bow_features_names1.append(cnt1)

for cnt2 in vectorizer_states.get_feature_names() :
    bow_features_names1.append(cnt2)

for cnt3 in vectorizer_grade.get_feature_names() :
    bow_features_names1.append(cnt3)

for cnt4 in vectorizer_teacher.get_feature_names() :
    bow_features_names1.append(cnt4)
```

In [120]:

```
len(bow_features_names1)
```

Out[120]:

```
100
```

In [121]:

```
bow_features_names1.append("price")

bow_features_names1.append("quantity")


bow_features_names1.append("prev_proposed_projects")

bow_features_names1.append("title_word_count")

bow_features_names1.append("essay_word_count")

len(bow_features_names1)
```

Out[121]:

```
105
```

In [122]:

```
for cnt5 in vectorizer_bow_title.get_feature_names() :
    bow_features_names1.append(cnt5)
```

In [123]:

```
len(bow_features_names1)
```

Out[123]:

```
2210
```

In [124]:

```
for cnt6 in vectorizer_bow_essay.get_feature_names() :
    bow_features_names1.append(cnt6)
```

In [125]:

```
len(bow_features_names1)
```

Out[125]:

```
7210
```

top 30 positive features BOW

NOTE

NOTE

- Using argsort by default it sorts in ascending order,but we need sorted log probabilities in desc order.
- While sorting the log probabilities in ascending order we get least imp features because just for an example say f1 has prob 16 and f2 has prob -14 .The actual prob of f1 would be exp(-16) and actual prob of f2 would be exp(-14). Clearly f2 has higher prob so it is an imp feature

In [ ]:

```
# To use argsort for descending order
# Code snippet taken from https://stackoverflow.com/questions/16486252/is-it-possible-to-use-argso
rt-in-descending-order
```

In [126]:

```
pos_class_prob_sorted = nb_bow.feature_log_prob_[1, :].argsort()[::-1][:7206]
for i in pos_class_prob_sorted[:30]:
    print(bow_features_names1[i])
```

```
price
essay_word_count
quantity
prev_proposed_projects
to
and
the
students
of
title_word_count
in
my
are
they
their
will
is
for
our
have
that
with
school
be
we
them
as
learning
on
classroom
```

30 negative features from BOW set1

In [ ]:

```
# To use argsort for descending order
# Code snippet taken from https://stackoverflow.com/questions/16486252/is-it-possible-to-use-argso
rt-in-descending-order
# using log_prob_ Code taken from https://scikit-
learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html
```

In [127]:

```
neg_class_prob_sorted = nb_bow.feature_log_prob_[0, :].argsort()[::-1][:7206]
for i in neg_class_prob_sorted[0:30]:
    print(bow_features_names1[i])
```

```
price
essay_word_count
quantity
to
```

```
and
the
prev_proposed_projects
students
of
title_word_count
in
my
are
they
their
will
is
that
our
for
have
with
school
be
we
them
learning
as
on
this
```

Summary

- Words like learn is present in negative class but not in positive class
- Few words are similar but their relative ordering is different between the two sets

tfidf top features

In [128]:

```
X_tr1 = hstack((categories_one_hot_train, sub_categories_one_hot_train,
school_state_categories_one_hot_train, project_grade_categories_one_hot_train,
teacher_prefix_categories_one_hot_train, price_train, quantity_train, prev_projects_train, title_wo
rd_count_train, essay_word_count_train, text_tfidf_train, title_tfidf_train)).tocsr()
X_te1 = hstack((categories_one_hot_test, sub_categories_one_hot_test,
school_state_categories_one_hot_test, project_grade_categories_one_hot_test,
teacher_prefix_categories_one_hot_test, price_test, quantity_test, prev_projects_test,
title_word_count_test, essay_word_count_test, text_tfidf_test, title_tfidf_test)).tocsr()
X_cr1 = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,
school_state_categories_one_hot_cv, project_grade_categories_one_hot_cv,
teacher_prefix_categories_one_hot_cv, price_cv, quantity_cv, prev_projects_cv, title_word_count_cv,
essay_word_count_cv, text_tfidf_cv, title_tfidf_cv)).tocsr()
```

In [129]:

```
X_tr1.shape
```

Out[129]:

```
(49041, 7210)
```

In [131]:

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

nb_tfidf.fit(X_tr, y_train)
```

Out[131]:

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

In [132]:

```
tfidf_features_names = []
```

In [133]:

```
for ct1 in vectorizer_proj.get_feature_names() :
    tfidf_features_names.append(ct1)

for ct2 in vectorizer_sub_proj.get_feature_names() :
    tfidf_features_names.append(ct2)

for ct3 in vectorizer_states.get_feature_names() :
    tfidf_features_names.append(ct3)


for ct4 in vectorizer_grade.get_feature_names() :
    tfidf_features_names.append(ct4)


for ct5 in vectorizer_teacher.get_feature_names() :
    tfidf_features_names.append(ct5)
```

In [134]:

```
len(tfidf_features_names)
```

Out[134]:

```
100
```

In [135]:

```
tfidf_features_names.append("price")

tfidf_features_names.append("quantity")

tfidf_features_names.append("prev_proposed_projects")

tfidf_features_names.append("title_word_count")

tfidf_features_names.append("essay_word_count")

for ct6 in vectorizer_tfidf_titles.get_feature_names() :
    tfidf_features_names.append(ct6)

for ct7 in vectorizer_tfidf_essay.get_feature_names() :
    tfidf_features_names.append(ct7)
```

In [136]:

```
len(tfidf_features_names)
```

Out[136]:

```
7210
```

positive features of tfidf

In [137]:

```
pos_class_prob_sorted_tfidf = nb_tfidf.feature_log_prob_[1, :].argsort()[::-1][:7210]
for i in pos_class_prob_sorted_tfidf[0:30]:
    print(tfidf_features_names[i])
```

```
price
essay_word_count
quantity
prev_proposed_projects
to
```

```
and
the
students
of
title_word_count
in
my
are
they
their
will
is
for
our
have
that
with
school
be
we
them
as
learning
on
classroom
```

Negative features from Tfidf

In [138]:

```python
neg_class_prob_sorted_tfidf = nb_tfidf.feature_log_prob_[0, :].argsort()[::-1][:7210]
for i in neg_class_prob_sorted_tfidf[0:30]:
    print(tfidf_features_names[i])
```

```
price
essay_word_count
quantity
to
and
the
prev_proposed_projects
students
of
title_word_count
in
my
are
they
their
will
is
that
our
for
have
with
school
be
we
them
learning
as
on
this
```

summary

- Again important features appear similar at first glance but actually there are some differences compared to set1
- relative ordering is different between the two sets

conclusions

In [139]:

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

from prettytable import PrettyTable

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

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

x.add_row(["BOW", "Naive Bayes", 0.001, 0.63])
x.add_row(["TFIDF", "Naive Bayes", 1e-05, 0.62])

print(x)
```

```
+------------+-------------+-----------------------+-----------+
| Vectorizer |    Model    | Alpha:Hyper Parameter |  Test AUC |
+------------+-------------+-----------------------+-----------+
|    BOW     | Naive Bayes |         0.001         |    0.63   |
|   TFIDF    | Naive Bayes |         1e-05         |    0.62   |
+------------+-------------+-----------------------+-----------+
```

- We conclude that Naive bayes gives better AUC than KNN
- also it is very fast as compared to KNN.
- naive Bayes is super interpretable because of probability values, we can get feature importance very easily as seen above
- There is strong possibility that Naive bayes can overfit if alpha has not been found properly

difference between fit(),transform(),fit_transform()

- To center the data (make it have zero mean and unit standard error), you subtract the mean and then divide the result by the standard deviation.
- fit() just calculates the parameters (e.g. mu and sigma in case of StandardScaler) and saves them as an internal objects state. Afterwards, you can call its transform() method to apply the transformation to a particular set of examples
- for egs fit() function happens only on training data while transform () involves changing the values by keeping mu and sigma in calculation x'= ((x-mu)/sigma))
- Using fix_transform(), we join these two steps and is used for the initial fitting of parameters on the training set x, but it also returns a transformed x′. Internally, it just calls first fit() and then transform() on the same data.
- generally fit_transform() should be applied on train data,and not on cv and test data,once fit has been done then we can use transform () on cv and test data

In [ ]: