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	Descri
`project_id`	A unique identifier for the proposed project. **Example:** `p0`
`project_title`	Title of the project. **Exam** • `Art Will Make You Ha` • `First Grade`
`project_grade_category`	Grade level of students for which the project is targeted. One of the foll enumerated va • `Grades Pr` • `Grades` • `Grades` • `Grades`

Feature	Descri
`project_subject_categories`	One or more (comma-separated) subject categories for the project fro following enumerated list of va • `Applied Lear` • `Care & Hu` • `Health & Sp` • `History & Ci` • `Literacy & Lang` • `Math & Sci` • `Music & The` • `Special N` • `Wa` **Exam** • `Music & The` • `Literacy & Language, Math & Sci`
`school_state`	State where school is located (Two-letter U.S. postal (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_co **Exampl**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the pr **Exam** • `Lite` • `Literature & Writing, Social Scie`
`project_resource_summary`	An explanation of the resources needed for the project. **Exan** • `My students need hands on literacy materials to man sensory ne`
`project_essay_1`	First application e
`project_essay_2`	Second application e
`project_essay_3`	Third application e
`project_essay_4`	Fourth application e
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04 12:43:56`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Exan** `bdf8baa8fedef6bfeec7ae4ff1c1`
`teacher_prefix`	Teacher's title. One of the following enumerated va • • • • • • `Teac`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same tea **Exam**

* 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`

Feature	Description
`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
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.
```

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)
```

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

In [4]:

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

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

Out[4]:

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

In [5]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:

In [6]:

```
project_grade_category = []

for i in range(len(project_data)):
    a = project_data["project_grade_category"][i].replace(" ", "_")
    project_grade_category.append(a)

project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data["project_grade_category"] = project_grade_category
project_data.head(5)
```

Out[6]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

1.2 preprocessing of `project_subject_categories`

In [7]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# 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", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [8]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# 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", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        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]))
```

Feature "Number of Words in Title"

In [9]:

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

project_data["title_word_count"] = title_word_count
project_data.head(5)
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

1.3 Text preprocessing

In [10]:

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

In [11]:

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

Out[11]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:

Number of Words in Essay

In [12]:

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

project_data["essay_word_count"] = essay_word_count

project_data.head(5)
```

Out[12]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

Type *Markdown* and LaTeX: α^2

In [13]:

```
# train test split

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_
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=
```

In [14]:

```
# train test split

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_
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=
```

In [15]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classroom
as well as the STEM journals, which my students really enjoyed.  I would lov
e to implement more of the Lakeshore STEM kits in my classroom for the next
school year as they provide excellent and engaging STEM lessons.My students
come from a variety of backgrounds, including language and socioeconomic sta
tus.  Many of them don't have a lot of experience in science and engineering
and these kits give me the materials to provide these exciting opportunities
for my students.Each month I try to do several science or STEM/STEAM project
s.  I would use the kits and robot to help guide my science instruction in e
ngaging and meaningful ways.  I can adapt the kits to my current language ar
ts pacing guide where we already teach some of the material in the kits like
tall tales (Paul Bunyan) or Johnny Appleseed.  The following units will be t
aught in the next school year where I will implement these kits: magnets, mo
tion, sink vs. float, robots.  I often get to these units and don't know If
I am teaching the right way or using the right materials.    The kits will g
ive me additional ideas, strategies, and lessons to prepare my students in s
cience.It is challenging to develop high quality science activities.  These
kits give me the materials I need to provide my students with science activi
ties that will go along with the curriculum in my classroom.  Although I hav
e some things (like magnets) in my classroom, I don't know how to use them e
ffectively.  The kits will provide me with the right amount of materials and
show me how to use them in an appropriate way.
==================================================
I teach high school English to students with learning and behavioral disabil
ities. My students all vary in their ability level. However, the ultimate go
al is to increase all students literacy levels. This includes their reading,
writing, and communication levels.I teach a really dynamic group of student
s. However, my students face a lot of challenges. My students all live in po
verty and in a dangerous neighborhood. Despite these challenges, I have stud
ents who have the the desire to defeat these challenges. My students all hav
e learning disabilities and currently all are performing below grade level.
My students are visual learners and will benefit from a classroom that fulfi
lls their preferred learning style.The materials I am requesting will allow
my students to be prepared for the classroom with the necessary supplies.  T
oo often I am challenged with students who come to school unprepared for cla
ss due to economic challenges.  I want my students to be able to focus on le
arning and not how they will be able to get school supplies.  The supplies w
ill last all year.  Students will be able to complete written assignments an
d maintain a classroom journal.  The chart paper will be used to make learni
ng more visual in class and to create posters to aid students in their learn
ing.  The students have access to a classroom printer.  The toner will be us
ed to print student work that is completed on the classroom Chromebooks.I wa
nt to try and remove all barriers for the students learning and create oppor
tunities for learning. One of the biggest barriers is the students not havin
g the resources to get pens, paper, and folders. My students will be able to
increase their literacy skills because of this project.
==================================================
\"Life moves pretty fast. If you don't stop and look around once in awhile,
you could miss it.\"  from the movie, Ferris Bueller's Day Off.  Think bac
k...what do you remember about your grandparents?  How amazing would it be t
o be able to flip through a book to see a day in their lives?My second grade
rs are voracious readers! They love to read both fiction and nonfiction book
s.  Their favorite characters include Pete the Cat, Fly Guy, Piggie and Elep
hant, and Mercy Watson. They also love to read about insects, space and plan
ts. My students are hungry bookworms! My students are eager to learn and rea
d about the world around them. My kids love to be at school and are like lit
tle sponges absorbing everything around them. Their parents work long hours
and usually do not see their children. My students are usually cared for by
their grandparents or a family friend. Most of my students do not have someo
ne who speaks English at home. Thus it is difficult for my students to acqui
re language.Now think forward... wouldn't it mean a lot to your kids, nieces
or nephews or grandchildren, to be able to see a day in your life today 30 y
ears from now? Memories are so precious to us and being able to share these
memories with future generations will be a rewarding experience.  As part of
our social studies curriculum, students will be learning about changes over
time.  Students will be studying photos to learn about how their community h
as changed over time.  In particular, we will look at photos to study how th
e land, buildings, clothing, and schools have changed over time.  As a culmi
nating activity, my students will capture a slice of their history and prese
rve it through scrap booking. Key important events in their young lives will
be documented with the date, location, and names.   Students will be using p
hotos from home and from school to create their second grade memories.   The
ir scrap books will preserve their unique stories for future generations to
enjoy.Your donation to this project will provide my second graders with an o
pportunity to learn about social studies in a fun and creative manner.  Thro
ugh their scrapbooks, children will share their story with others and have a
historical document for the rest of their lives.
==================================================
\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smalles
t students with the biggest enthusiasm for learning. My students learn in ma
ny different ways using all of our senses and multiple intelligences. I use
a wide range of techniques to help all my students succeed. \r\nStudents in
my class come from a variety of different backgrounds which makes for wonder
ful sharing of experiences and cultures, including Native Americans.\r\nOur
school is a caring community of successful learners which can be seen throug
h collaborative student project based learning in and out of the classroom.
Kindergarteners in my class love to work with hands-on materials and have ma
ny different opportunities to practice a skill before it is mastered. Having
the social skills to work cooperatively with friends is a crucial aspect of
the kindergarten curriculum.Montana is the perfect place to learn about agri
culture and nutrition. My students love to role play in our pretend kitchen
in the early childhood classroom. I have had several kids ask me, \"Can we t
ry cooking with REAL food?\" I will take their idea and create \"Common Core
Cooking Lessons\" where we learn important math and writing concepts while c
ooking delicious healthy food for snack time. My students will have a ground
ed appreciation for the work that went into making the food and knowledge of
where the ingredients came from as well as how it's healthy for their bodie
s. This project would expand our learning of nutrition and agricultural cook
ing recipes by having us peel our own apples to make homemade applesauce, ma
ke our own bread, and mix up healthy plants from our classroom garden in the
spring. We will also create our own cookbooks to be printed and shared with
families. \r\nStudents will gain math and literature skills as well as a lif
e long enjoyment for healthy cooking.nannan
==================================================

My classroom consists of twenty-two amazing sixth graders from different c
ultures and backgrounds. They are a social bunch who enjoy working in part
ners and working with groups. They are hard-working and eager to head to m
```

```
iddle school next year. My job is to get them ready to make this transitio
n and make it as smooth as possible. In order to do this, my students need
to come to school every day and feel safe and ready to learn. Because they
are getting ready to head to middle school, I give them lots of choice- ch
oice on where to sit and work, the order to complete assignments, choice o
f projects, etc. Part of the students feeling safe is the ability for them
to come into a welcoming, encouraging environment. My room is colorful and
the atmosphere is casual. I want them to take ownership of the classroom b
ecause we ALL share it together. Because my time with them is limited, I w
ant to ensure they get the most of this time and enjoy it to the best of t
heir abilities.Currently, we have twenty-two desks of differing sizes, yet
the desks are similar to the ones the students will use in middle school.
We also have a kidney table with crates for seating. I allow my students t
o choose their own spots while they are working independently or in group
s. More often than not, most of them move out of their desks and onto the
crates. Believe it or not, this has proven to be more successful than maki
ng them stay at their desks! It is because of this that I am looking towar
d the “Flexible Seating” option for my classroom.\r\n The students look fo
rward to their work time so they can move around the room. I would like to
get rid of the constricting desks and move toward more “fun” seating optio
ns. I am requesting various seating so my students have more options to si
t. Currently, I have a stool and a papasan chair I inherited from the prev
ious sixth-grade teacher as well as five milk crate seats I made, but I wo
uld like to give them more options and reduce the competition for the “goo
d seats”. I am also requesting two rugs as not only more seating options b
ut to make the classroom more welcoming and appealing. In order for my stu
dents to be able to write and complete work without desks, I am requesting
a class set of clipboards. Finally, due to curriculum that requires groups
to work together, I am requesting tables that we can fold up when we are n
ot using them to leave more room for our flexible seating options.\r\nI kn
ow that with more seating options, they will be that much more excited abo
ut coming to school! Thank you for your support in making my classroom one
students will remember forever!nannan
==================================================
```

In [16]:

```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 [17]:

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

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smalle
st students with the biggest enthusiasm for learning. My students learn in m
any different ways using all of our senses and multiple intelligences. I use
a wide range of techniques to help all my students succeed. \r\nStudents in
my class come from a variety of different backgrounds which makes for wonder
ful sharing of experiences and cultures, including Native Americans.\r\nOur
school is a caring community of successful learners which can be seen throug
h collaborative student project based learning in and out of the classroom.
Kindergarteners in my class love to work with hands-on materials and have ma
ny different opportunities to practice a skill before it is mastered. Having
the social skills to work cooperatively with friends is a crucial aspect of
the kindergarten curriculum.Montana is the perfect place to learn about agri
culture and nutrition. My students love to role play in our pretend kitchen
in the early childhood classroom. I have had several kids ask me, \"Can we t
ry cooking with REAL food?\" I will take their idea and create \"Common Core
Cooking Lessons\" where we learn important math and writing concepts while c
ooking delicious healthy food for snack time. My students will have a ground
ed appreciation for the work that went into making the food and knowledge of
where the ingredients came from as well as how it is healthy for their bodie
s. This project would expand our learning of nutrition and agricultural cook
ing recipes by having us peel our own apples to make homemade applesauce, ma
ke our own bread, and mix up healthy plants from our classroom garden in the
spring. We will also create our own cookbooks to be printed and shared with
families. \r\nStudents will gain math and literature skills as well as a lif
e long enjoyment for healthy cooking.nannan
==================================================
```

In [18]:

```
# \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)
```

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the smallest
students with the biggest enthusiasm for learning. My students learn in many
different ways using all of our senses and multiple intelligences. I use a w
ide range of techniques to help all my students succeed.   Students in my cl
ass come from a variety of different backgrounds which makes for wonderful s
haring of experiences and cultures, including Native Americans.  Our school 
is a caring community of successful learners which can be seen through colla
borative student project based learning in and out of the classroom. Kinderg
arteners in my class love to work with hands-on materials and have many diff
erent opportunities to practice a skill before it is mastered. Having the so
cial skills to work cooperatively with friends is a crucial aspect of the ki
ndergarten curriculum.Montana is the perfect place to learn about agricultur
e and nutrition. My students love to role play in our pretend kitchen in the
early childhood classroom. I have had several kids ask me,  Can we try cooki
ng with REAL food?  I will take their idea and create  Common Core Cooking L
essons  where we learn important math and writing concepts while cooking del
icious healthy food for snack time. My students will have a grounded appreci
ation for the work that went into making the food and knowledge of where the
ingredients came from as well as how it is healthy for their bodies. This pr
oject would expand our learning of nutrition and agricultural cooking recipe
s by having us peel our own apples to make homemade applesauce, make our own
bread, and mix up healthy plants from our classroom garden in the spring. We
will also create our own cookbooks to be printed and shared with families. 
Students will gain math and literature skills as well as a life long enjoyme
nt for healthy cooking.nannan
```

In [19]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
 A person is a person no matter how small Dr Seuss I teach the smallest stud
ents with the biggest enthusiasm for learning My students learn in many diff
erent ways using all of our senses and multiple intelligences I use a wide r
ange of techniques to help all my students succeed Students in my class come
from a variety of different backgrounds which makes for wonderful sharing of
experiences and cultures including Native Americans Our school is a caring c
ommunity of successful learners which can be seen through collaborative stud
ent project based learning in and out of the classroom Kindergarteners in my
class love to work with hands on materials and have many different opportuni
ties to practice a skill before it is mastered Having the social skills to w
ork cooperatively with friends is a crucial aspect of the kindergarten curri
culum Montana is the perfect place to learn about agriculture and nutrition
My students love to role play in our pretend kitchen in the early childhood
classroom I have had several kids ask me Can we try cooking with REAL food I
will take their idea and create Common Core Cooking Lessons where we learn i
mportant math and writing concepts while cooking delicious healthy food for
snack time My students will have a grounded appreciation for the work that w
ent into making the food and knowledge of where the ingredients came from as
well as how it is healthy for their bodies This project would expand our lea
rning of nutrition and agricultural cooking recipes by having us peel our ow
n apples to make homemade applesauce make our own bread and mix up healthy p
lants from our classroom garden in the spring We will also create our own co
okbooks to be printed and shared with families Students will gain math and l
iterature skills as well as a life long enjoyment for healthy cooking nannan
```

In [20]:

```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'd
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            'won', "won't", 'wouldn', "wouldn't"]
```

Preprocessed Train data (Essay)

In [21]:

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

```
100%|███████████████████████████████████████████████████████████████████████
████| 49041/49041 [00:57<00:00, 854.56it/s]
```

In [22]:

```
# after preprocesing
preprocessed_essays_train[20000]
```

Out[22]:

```
'teach title l school east san jose majority students first generation ameri
cans hardworking immigrant parents eighty percent students school receiving
free reduced priced lunch teach wonderful group kindergarten students studen
ts hard working enthusiastic learners easily motivated enjoy attending schoo
l every day always excited try learn new concepts students inspire every day
better teacher students enjoy building little hands like construct towers bu
ildings high currently wooden blocks disposal need ways express show amazing
engineering skills wooden blocks great easily topple students get sad frustr
ated building collapse finish magnetic shapes request help solve problem sha
pes magnets make building easy young kids magnetic shapes allows students de
sign create buildings time learn symmetry area perimeter magnetic shapes hel
p provide students engineering skills successful nannan'
```

Preprocessed Test data (Essay)

In [23]:

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

```
100%|███████████████████████████████████████████████████████████████████████
████| 36052/36052 [00:41<00:00, 861.52it/s]
```

In [24]:

```
preprocessed_essays_test[0]
```

Out[24]:

```
'classroom consists 4th graders driven learn grow always passion mine work u
rban students learn education anything possible getting good education chall
enging also learn challenges help us grow better students dedicated mantra i
mpress everyday challenging things accomplish students passionate innovative
intelligent motivated love engage hands learning experiences use everything 
real world experiences virtual experiences enhance educational opportunities
always marvel wonderful things create virtually using art medias using scien
tific vex kits create innovative solutions problems important us school prov
ide many extra curricular activities urban students love give opportunity ge
t involved many clubs sports provide matters us give children chance explore
opportunities outside classroom leaders collaborators always looking ways ge
t students active giving opportunity involved team sports great way achieve 
goal would using equipment create inter school soccer league plan create tea
ms allow kids compete classmates training competing public park school not r
ight space us play makes portable equipment much important us nannan'
```

Preprocessed Cross Validation data (essay)

In [25]:

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

```
100%|████████████████████████████████████████████████████████████████████████
████| 24155/24155 [00:29<00:00, 823.51it/s]
```

In [26]:

```
preprocessed_essays_cv[0]
```

Out[26]:

```
'work military city school labeled highest poverty entire school receives fr
ee breakfast lunch every day majority kindergartners not go pre k first year
school start year basic letter identification end year reading ready head fi
rst grade students hard working enthusiastic appreciative students not easy
home lives important create safe place students learn classroom community li
ke family every day work together best fun try hardest provide students best
first year school researching personally seeing difference flexible seating
make classroom slowly started bringing incredible new idea room started boun
cy balls stools see would positive effect ability focus also remain active s
chool day students loved new seating excited opportunity use clear comfort e
nhances concentration eagerness school whole problem not enough set balance
balls wobble stools wiggle cushions help meet needs kindergarteners allowing
move wiggle learn additionally help promote positive learning environment al
lowing choice students nannan'
```

1.4 Preprocessing of `project_title`

Preprocessing of Project Title for Train data

In [27]:

```
# similarly you can preprocess the titles also
preprocessed_titles_train = []

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

```
100%|████████████████████████████████████████████████████████████████████████
█| 49041/49041 [00:02<00:00, 18017.92it/s]
```

In [28]:

```
preprocessed_titles_train[0]
```

Out[28]:

```
'the painters studio'
```

Preprocessing of Project Title for Test data

In [29]:

```
preprocessed_titles_test = []

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

```
100%|████████████████████████████████████████████████████████████
█| 36052/36052 [00:01<00:00, 18396.84it/s]
```

In [30]:

```
preprocessed_titles_test[0]
```

Out[30]:

```
'windlake academy soccer'
```

Preprocessing of Project Title for CV data

In [31]:

```
preprocessed_titles_cv = []

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

```
100%|████████████████████████████████████████████████████████████
█| 24155/24155 [00:01<00:00, 19614.63it/s]
```

In [32]:

```
preprocessed_titles_cv[0]
```

Out[32]:

```
'wiggle while you work'
```

1.5 Preparing data for models

In [33]:

```
project_data.columns
```

Out[33]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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_approve
d',
       'project_grade_category', '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

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/ (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/)

In [34]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_proj = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
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.shap
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)
```

In [35]:

```
# we use count vectorizer to convert the values into one
vectorizer_sub_proj = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercas
vectorizer_sub_proj.fit(X_train['clean_subcategories'].values)

sub_categories_one_hot_train = vectorizer_sub_proj.transform(X_train['clean_subcategories']
sub_categories_one_hot_test = vectorizer_sub_proj.transform(X_test['clean_subcategories'].v
sub_categories_one_hot_cv = vectorizer_sub_proj.transform(X_cv['clean_subcategories'].value


print(vectorizer_sub_proj.get_feature_names())

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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducat
ion', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Characte
rEducation', '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)
```

In [36]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())
```

In [37]:

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

In [38]:

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

vectorizer_states = CountVectorizer(vocabulary=list(sorted_school_state_cat_dict.keys()), l
vectorizer_states.fit(X_train['school_state'].values)

school_state_categories_one_hot_train = vectorizer_states.transform(X_train['school_state']
school_state_categories_one_hot_test = vectorizer_states.transform(X_test['school_state'].v
school_state_categories_one_hot_cv = vectorizer_states.transform(X_cv['school_state'].value

print(vectorizer_states.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",school_state_categories_one_h
print("Shape of matrix of Test data after one hot encoding ",school_state_categories_one_ho
print("Shape of matrix of Cross Validation data after one hot encoding ",school_state_categ
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'H
I', 'DC', 'NM', 'KS', 'IA', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV',
'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'MA', 'L
A', '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)
```

In [39]:

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

In [40]:

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

In [41]:

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

vectorizer_grade = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()), l
vectorizer_grade.fit(X_train['project_grade_category'].values)

project_grade_categories_one_hot_train = vectorizer_grade.transform(X_train['project_grade_
project_grade_categories_one_hot_test = vectorizer_grade.transform(X_test['project_grade_ca
project_grade_categories_one_hot_cv = vectorizer_grade.transform(X_cv['project_grade_catego

print(vectorizer_grade.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",project_grade_categories_one_
print("Shape of matrix of Test data after one hot encoding ",project_grade_categories_one_h
print("Shape of matrix of Cross Validation data after one hot encoding ",project_grade_cate
```

```
['Grades_9-12', 'Grades_6-8', 'Grades_3-5', 'Grades_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)
```

In [42]:

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

In [43]:

```
teacher_prefix_cat_dict = dict(my_counter)
sorted_teacher_prefix_cat_dict = dict(sorted(teacher_prefix_cat_dict.items(), key=lambda kv
```

In [44]:

```
## we use count vectorizer to convert the values into one hot encoded features
## Unlike the previous Categories this category returns a
## ValueError: np.nan is an invalid document, expected byte or unicode string.
## The link below explains h0w to tackle such discrepancies.
## https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-

vectorizer_teacher = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.keys())
vectorizer_teacher.fit(X_train['teacher_prefix'].values.astype("U"))

teacher_prefix_categories_one_hot_train = vectorizer_teacher.transform(X_train['teacher_pre
teacher_prefix_categories_one_hot_test = vectorizer_teacher.transform(X_test['teacher_prefi
teacher_prefix_categories_one_hot_cv = vectorizer_teacher.transform(X_cv['teacher_prefix'].

print(vectorizer_teacher.get_feature_names())

print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_train.sha
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_test.shap
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_cv.shape)
```

```
['nan', 'Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encoding  (49041, 6)
Shape of matrix after one hot encoding  (36052, 6)
Shape of matrix after one hot encoding  (24155, 6)
```

In [ ]:

```

```

1.5.2 Vectorizing Text data

a) Bag of words Train Data (Essays)

In [45]:

```
# We are considering only the words which appeared in at least 10 documents(rows or project
vectorizer_bow_essay = CountVectorizer(min_df=10)

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, 12035)
```

b) Bag of words Test Data (Essays)

In [46]:

```
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, 12035)
```

c) Bag of words CV Data (Essays)

In [47]:

```
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, 12035)
```

d) Bag of words train Data (Titles)

In [48]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_bow_title = CountVectorizer(min_df=10)

vectorizer_bow_title.fit(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, 2091)
```

e) Bag of words Test Data (Titles)

In [49]:

```
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, 2091)
```

f) Bag of words Data (Titles)

In [50]:

```
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, 2091)
```

1.5.2.2 TFIDF vectorizer

a) TFIDF vectorizer Train Data (Essays)

In [51]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
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, 12035)
```

b) TFIDF vectorizer Test Data (Essays)

In [52]:

```
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, 12035)
```

c) TFIDF vectorizer CV Data (Essays)

In [53]:

```
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, 12035)
```

c) TFIDF vectorizer Train Data (Titles)

In [54]:

```
vectorizer_tfidf_titles = TfidfVectorizer(min_df=10)

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, 2091)
```

d) TFIDF vectorizer Test Data (Titles)

In [55]:

```
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, 2091)
```

e) TFIDF vectorizer CV Data (Titles)

In [56]:

```
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, 2091)
```

1.5.3 Vectorizing Numerical features

a) Price

In [57]:

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

In [58]:

```
# 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 [72]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikitlearn.org/stable/modules/generated/sklearn.preproc
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
price_scalar = MinMaxScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standarddevi
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
price_train = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
price_train
# Now standardize the data with above maen and variance.
price_test = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
price_test
# Now standardize the data with above maen and variance.
price_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_cv
```

Out[72]:

```
array([[0.0072152 ],
       [0.00113419],
       [0.02052541],
       ...,
       [0.0108548 ],
       [0.01030271],
       [0.03199231]])
```

In [73]:

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

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

b) Quantity

In [76]:

```
price_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
quantity_train = price_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
quantity_train
# Now standardize the data with above maen and variance.
quantity_cv = price_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
quantity_cv
# Now standardize the data with above maen and variance.
quantity_test = price_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
quantity_test

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

```
After vectorizations
<built-in method reshape of numpy.ndarray object at 0x000001ACD97564E0> (490
41,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

c) Number of Projects previously proposed by Teacher

In [77]:

```
price_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
prev_projects_train = price_scalar.transform(X_train['teacher_number_of_previously_posted_p
prev_projects_train
# Now standardize the data with above maen and variance.
prev_projects_cv = price_scalar.transform(X_cv['teacher_number_of_previously_posted_project
prev_projects_cv
# Now standardize the data with above maen and variance.
prev_projects_test = price_scalar.transform(X_test['teacher_number_of_previously_posted_pro
prev_projects_test

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)
```

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

d) Count of Words in the Title

In [78]:

```
price_scalar.fit(X_train['title_word_count'].values.reshape(-1,1)) # finding the mean and s
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
title_word_count_train = price_scalar.transform(X_train['title_word_count'].values.reshape(
title_word_count_train
# Now standardize the data with above maen and variance.
title_word_count_cv = price_scalar.transform(X_cv['title_word_count'].values.reshape(-1, 1)
title_word_count_cv
# Now standardize the data with above maen and variance.
title_word_count_test = price_scalar.transform(X_test['title_word_count'].values.reshape(-1
title_word_count_test


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)
```

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

b) Count of Words in the Essay

In [79]:

```
price_scalar.fit(X_train['essay_word_count'].values.reshape(-1,1)) # finding the mean and s
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
essay_word_count_train = price_scalar.transform(X_train['essay_word_count'].values.reshape(
essay_word_count_train
# Now standardize the data with above maen and variance.
essay_word_count_cv = price_scalar.transform(X_cv['essay_word_count'].values.reshape(-1, 1)
essay_word_count_cv
# Now standardize the data with above maen and variance.
essay_word_count_test = price_scalar.transform(X_test['essay_word_count'].values.reshape(-1
essay_word_count_test


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

In [80]:

```
essay_word_count_train
```

Out[80]:

```
array([[0.32631579],
       [0.48684211],
       [0.44736842],
       ...,
       [0.20526316],
       [0.36052632],
       [0.46315789]])
```

Assignment 4: Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

2. **The hyper paramter tuning(find best Alpha)**
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Feature importance**
    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding feature names

4. **Representation of results**
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
    - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
    - Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps. (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

5. **Conclusion**

- You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link (http://zetcode.com/python/prettytable/)

2. Naive Bayes

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

In [81]:

```
ain, quantity_train, prev_projects_train, title_word_count_train, essay_word_count_train, ti
uantity_test, prev_projects_test, title_word_count_test, essay_word_count_test, title_bow_te
prev_projects_cv, title_word_count_cv, essay_word_count_cv, title_bow_cv, text_bow_cv)).tocs
```

In [82]:

```
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, 14231) (49041,)
(24155, 14231) (24155,)
(36052, 14231) (36052,)
============================================================================
========================
```

In [83]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # 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 = 4900
    # 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

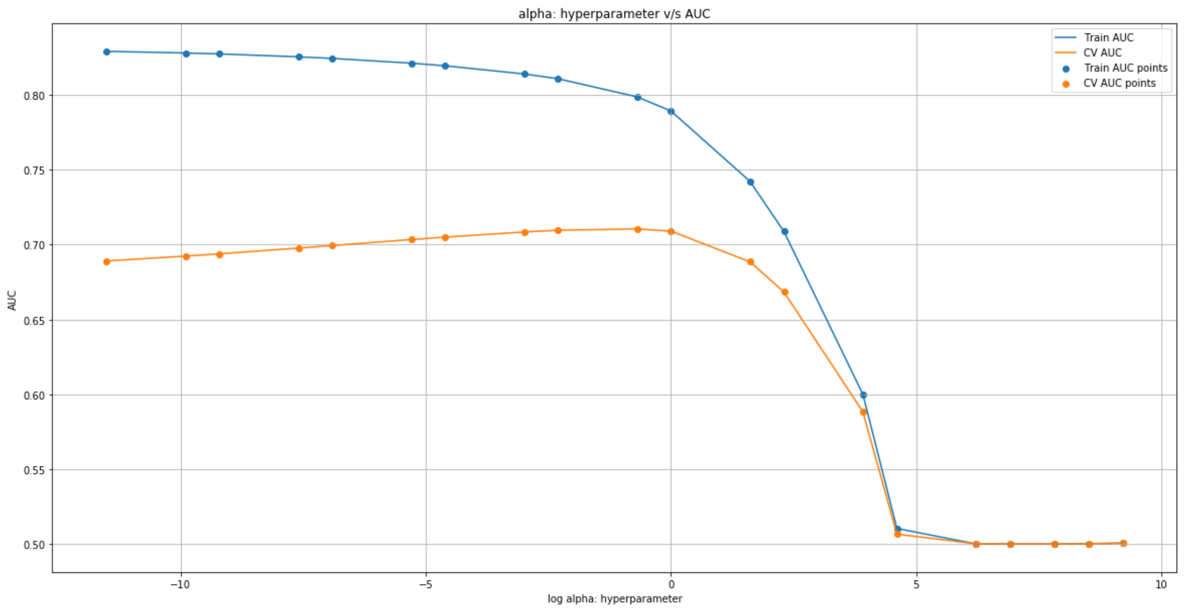
In [84]:

```
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, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 5

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 t
    # 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%|████████████████████████████████████████████████████████████████████████
██████████| 20/20 [00:10<00:00,  2.96it/s]
100%|████████████████████████████████████████████████████████████████████████
████████████████████| 20/20 [00:00<?, ?it/s]
```

In [85]:

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

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

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



B) Gridsearch-cv

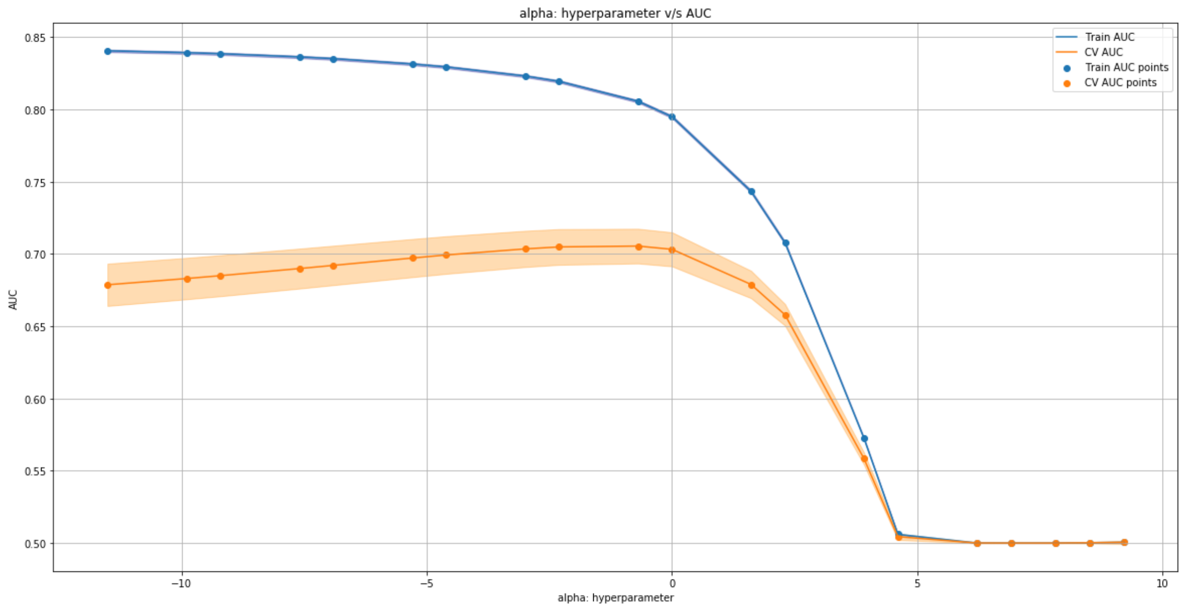
In [86]:

```
from sklearn.model_selection import GridSearchCV

nb = MultinomialNB()

parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5

clf = GridSearchCV(nb, parameters, cv= 10, scoring='roc_auc')

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']
```

In [87]:

```
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 5
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

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='

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()
plt.show()
```

```
100%|██████████████████████████████████████████████████████
███████████████| 20/20 [00:00<?, ?it/s]
```



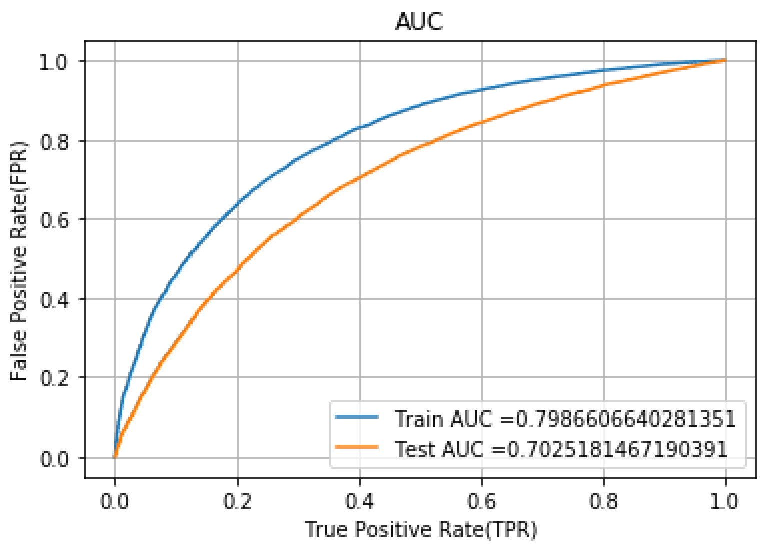
In [88]:

```
best_k_1 = 0.5
```

C) Train model using the best hyper-parameter value

In [89]:

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

nb_bow = MultinomialNB(alpha = best_k_1)

nb_bow.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# 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()
plt.show()
```



D) Confusion Matrix

In [90]:

```
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
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

Train Data

In [91]:

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

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999818661462 for threshold 0.118
[[ 3714  3712]
 [ 4722 36893]]
```

In [92]:

```
conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

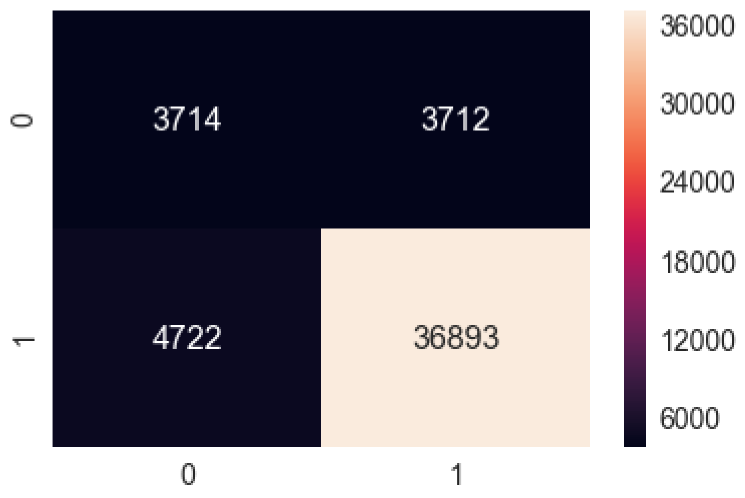
```
the maximum value of tpr*(1-fpr) 0.2499999818661462 for threshold 0.118
```

In [93]:

```
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[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac8c779588>
```



Test Data

In [94]:

```
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 0.565
[[ 2806  2653]
 [ 6957 23636]]
```

In [95]:

```
conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_threshc
```

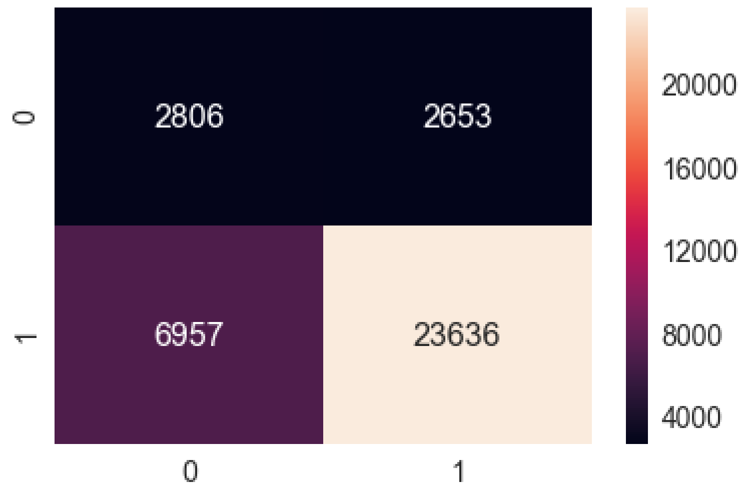
```
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 0.565
```

In [96]:

```
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[96]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac8d845400>
```



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

In [97]:

```
# 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_categor
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_categorie
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_categories_or
```

In [98]:

```
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)
```

```
Final Data matrix
(49041, 14231) (49041,)
(24155, 14231) (24155,)
(36052, 14231) (36052,)
```

Random Alpha Values

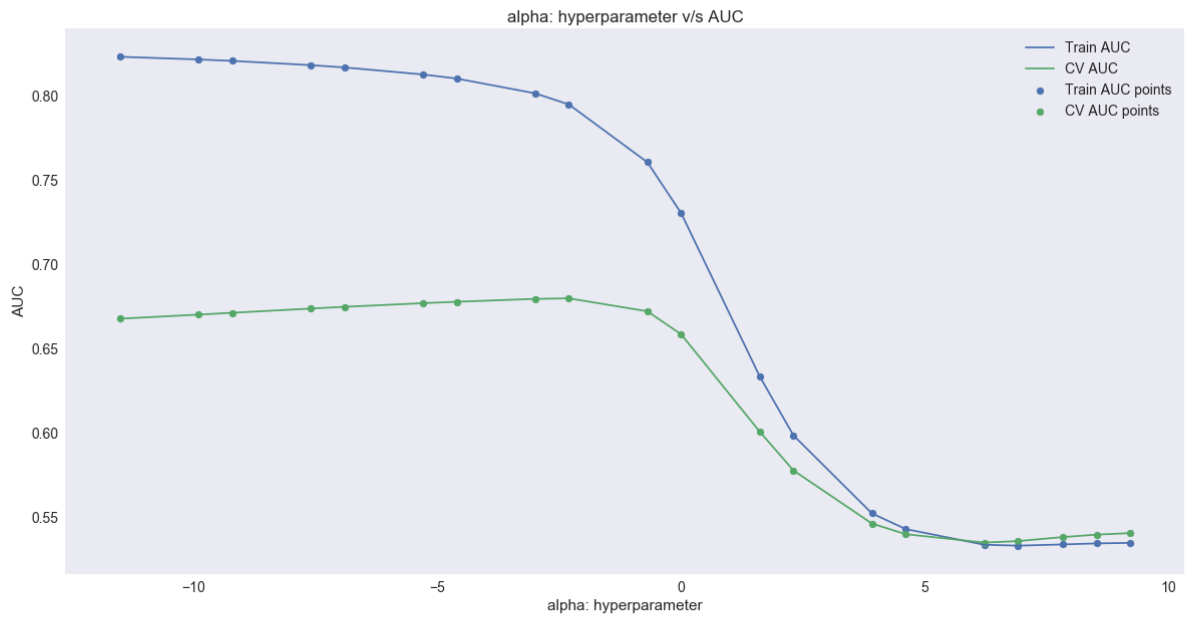
In [102]:

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

alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 5

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 t
    # 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%|██████████████████████████████████████████████████████████████████████
███████████| 20/20 [00:07<00:00,  3.23it/s]
100%|██████████████████████████████████████████████████████████████████████
███████████████████| 20/20 [00:00<?, ?it/s]
```

In [103]:

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



GridSearch CV

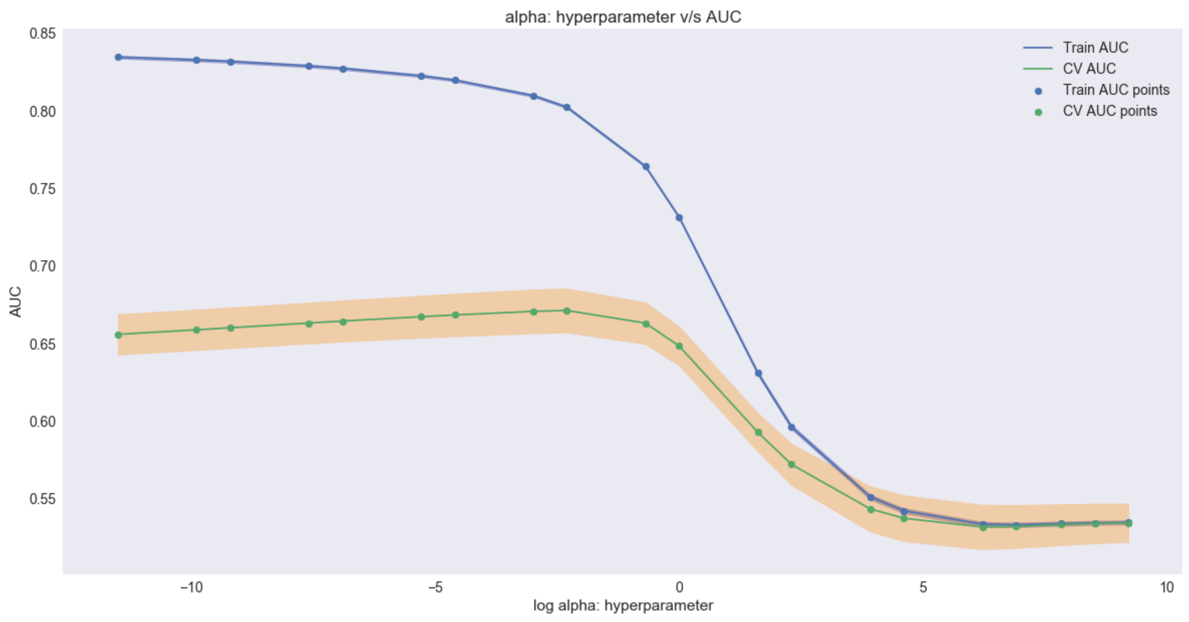
In [104]:

```
nb = MultinomialNB()

parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5

clf = GridSearchCV(nb, parameters, cv= 10, scoring='roc_auc')

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']
```

In [105]:

```
alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 5
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

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='

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


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

```
100%|████████████████████████████████████████████████████████
███████████████| 20/20 [00:00<?, ?it/s]
```



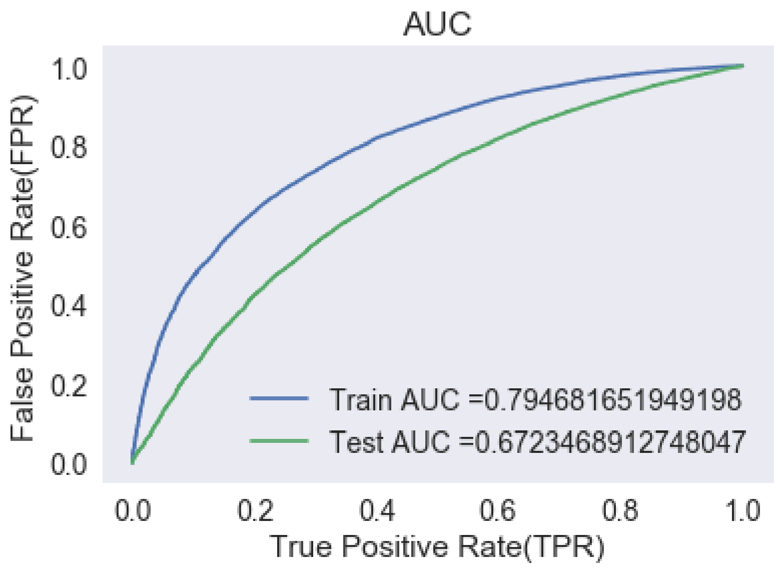
In [106]:

```
best_k_2 = 0.1
```

Train model using the best hyper-parameter value

In [107]:

```
nb_tfidf = MultinomialNB(alpha = best_k_2)

nb_tfidf.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# 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()
plt.show()
```



Confusion Matrix -Train data

In [108]:

```
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 0.757
[[ 3713  3713]
 [ 5345 36270]]
```

In [109]:

```
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

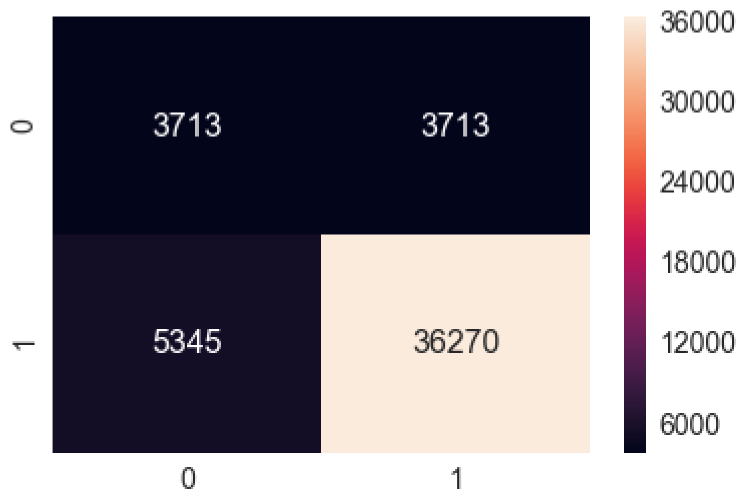
```
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.757
```

In [110]:

```
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[110]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac922a5518>
```



Test Data

In [111]:

```
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 0.812
[[ 2566  2893]
 [ 7107 23486]]
```

In [112]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_threshc
```

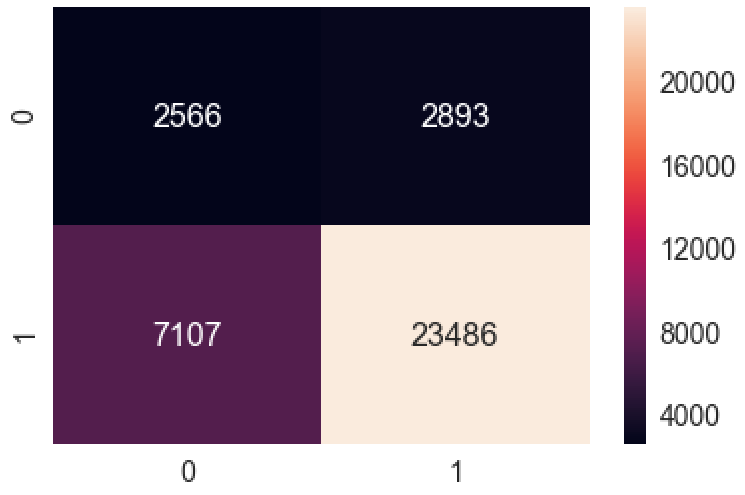
```
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 0.812
```

In [113]:

```
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[113]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1ac8ba2fb00>
```



Select best 10 features of both Positive and negative class for both the sets of data

Set 1: BOW

In [114]:

```
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_categor
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_categorie
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_categories_or
```

In [115]:

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

nb_bow.fit(X_tr, y_train)
```

Out[115]:

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

In [124]:

```
bow_features_probs = {}

for a in range(14231) :

    bow_features_probs[a] = nb_bow.feature_log_prob_[0,a]
```

In [125]:

```
len(bow_features_probs.values())
```

Out[125]:

```
14231
```

In [126]:

```
bow_features_names = []
for a in vectorizer_proj.get_feature_names() :
    bow_features_names.append(a)
```

In [127]:

```
for a in vectorizer_sub_proj.get_feature_names() :
    bow_features_names.append(a)
```

In [128]:

```
for a in vectorizer_states.get_feature_names() :
    bow_features_names.append(a)
```

In [129]:

```
for a in vectorizer_grade.get_feature_names() :
    bow_features_names.append(a)
```

In [130]:

```
for a in vectorizer_teacher.get_feature_names() :
    bow_features_names.append(a)
```

In [131]:

```
len(bow_features_names)
```

Out[131]:

```
100
```

In [132]:

```
bow_features_names.append("price")
bow_features_names.append("quantity")
bow_features_names.append("prev_proposed_projects")
bow_features_names.append("title_word_count")
bow_features_names.append("essay_word_count")
len(bow_features_names)
```

Out[132]:

```
105
```

In [133]:

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

In [134]:

```
len(bow_features_names)
```

Out[134]:

```
2196
```

In [135]:

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

In [136]:

```
len(bow_features_names)
```

Out[136]:

```
14231
```

30 Positive features from BOW model

In [137]:

```
# Code snippet taken from https://stackoverflow.com/questions/16486252/is-it-possible-to-us
pos_class_prob_sorted = nb_bow.feature_log_prob_[1, :].argsort()[::-1][:14207]
for i in pos_class_prob_sorted[:30]:
    print(bow_features_names[i])
```

```
students
school
learning
classroom
not
learn
help
many
nannan
reading
need
work
use
love
day
able
come
class
would
technology
also
books
skills
year
new
make
want
time
student
one
```

30 Negative features from BOW model

In [138]:

```
# To use argsort for descending order
# Code snippet taken from https://stackoverflow.com/questions/16486252/is-it-possible-to-us
# using log_prob_ Code taken from https://scikitlearn.org/stable/modules/generated/sklearn.
neg_class_prob_sorted = nb_bow.feature_log_prob_[0, :].argsort()[::-1][:14207]
for i in neg_class_prob_sorted[0:30]:
    print(bow_features_names[i])
```

```
students
school
learning
classroom
not
learn
help
nannan
many
need
work
come
reading
love
materials
skills
day
able
class
use
want
year
make
new
would
also
technology
student
one
time
```

In [139]:

```
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_categor
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_categorie
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_categories_or
```

In [141]:

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

nb_tfidf.fit(X_tr, y_train)
```

Out[141]:

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

In [144]:

```
tfidf_features_probs_neg = {}

for a in range(14231) :

    tfidf_features_probs_neg[a] = nb_tfidf.feature_log_prob_[0,a]
```

In [145]:

```
len(tfidf_features_probs_neg)
```

Out[145]:

```
14231
```

In [146]:

```
tfidf_features_names = []
for a in vectorizer_proj.get_feature_names() :
    tfidf_features_names.append(a)

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

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

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

for a in vectorizer_teacher.get_feature_names() :
    tfidf_features_names.append(a)

len(tfidf_features_names)
```

Out[146]:

```
100
```

In [147]:

```
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")
```

In [148]:

```
for a in vectorizer_tfidf_titles.get_feature_names() :
    tfidf_features_names.append(a)
```

In [149]:

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

30 Positive features from TFIDF model

In [150]:

```
pos_class_prob_sorted_tfidf = nb_tfidf.feature_log_prob_[1, :].argsort()[::-1][:14207]
for i in pos_class_prob_sorted_tfidf[0:30]:
    print(tfidf_features_names[i])
```

```
Literacy_Language
title_word_count
Math_Science
essay_word_count
Literacy
Mathematics
Literature_Writing
CA
process
Health_Sports
SpecialNeeds
SpecialNeeds
AppliedLearning
Health_Wellness
AppliedSciences
Music_Arts
NY
TX
VisualArts
FL
History_Civics
EnvironmentalScience
occupational
NC
Gym_Fitness
ESL
faced
way
IL
EarlyDevelopment
```

30 Negative features from TFIDF model

In [151]:

```
neg_class_prob_sorted_tfidf = nb_tfidf.feature_log_prob_[0, :].argsort()[::-1][:14207]
for i in neg_class_prob_sorted_tfidf[0:30]:
    print(tfidf_features_names[i])
```

```
Literacy_Language
Math_Science
title_word_count
essay_word_count
Mathematics
Literacy
Literature_Writing
SpecialNeeds
SpecialNeeds
process
CA
Health_Sports
AppliedLearning
AppliedSciences
Music_Arts
Health_Wellness
TX
VisualArts
FL
EnvironmentalScience
NY
occupational
History_Civics
NC
faced
EarlyDevelopment
Health_LifeScience
IL
Gym_Fitness
GA
```

3. Conclusions

In [152]:

```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

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

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

x.add_row(["BOW", "Naive Bayes", 0.5, 0.7])
x.add_row(["TFIDF", "Naive Bayes", 0.1, 0.67])

print(x)
```

```
+------------+-------------+-----------------------+------+
| Vectorizer |    Model    | Alpha:Hyper Parameter | AUC  |
+------------+-------------+-----------------------+------+
|    BOW     | Naive Bayes |          0.5          | 0.7  |
|   TFIDF    | Naive Bayes |          0.1          | 0.67 |
+------------+-------------+-----------------------+------+
```

In [ ]: