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**
`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 targete enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories f 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 ((https://en.wikipedia.org/wiki/List_of_U.S._state_abbr **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategori **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the projec • `My students need hands on literacy mater sensory needs!`

Feature	Description
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Ex** 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed pro bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix	Teacher's title. One of the following enumerated value • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted b **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_4:__ "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

import re
from tqdm import tqdm

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 the 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)
print('='*100)
project_data.head(2)
```

```
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]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_s
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL

Renaming and sorting the column name [project_submitted_datetime]

In [4]:

```
# 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/49702492/40840
39

#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)
```

In [5]:

```
# We can also do the same work by using pandas rename method.
project_data = project_data.rename(columns = {'project_submitted_datetime': 'Date'})

# Sorting the dataframe according to time
project_data['Date'] = pd.to_datetime(project_data['Date'])
project_data.sort_values(by = ['Date'], inplace = True)

project_data.head(2)
```

Out[5]:

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

In [6]:

```
print("Number of data points in resource data", resource_data.shape)
print('-'*50)
print("The attributes of resource data :", resource_data.columns.values)
print('='*100)
resource_data.head(2)
```

```
Number of data points in resource data (1541272, 4)
--------------------------------------------------
The attributes of resource data : ['id' 'description' 'quantity' 'price']
====================================================================================
==========================
```

Out[6]:

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

1.2 preprocessing of `project_subject_categories`

In [7]:

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

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

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

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

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

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/47
301924/4084039

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

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

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

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

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

1.3 Text preprocessing

In [9]:

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

In [10]:

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

Out[10]:

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

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

In [11]:

```
X = project_data.drop(['project_is_approved'],axis = 1)
y = project_data['project_is_approved'].values
```

In [12]:

```
X.head(2)
```

Out[12]:

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

In [13]:

```
y[10:]
```

Out[13]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [14]:

```
# Splitting into train, cv and test
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.33, stratify = y
)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size = 0.33, str
atify = y_train)
```

In [15]:

```
print('The shape of X_train & y_train: ',X_train.shape, y_train.shape)
print('The shape of X_cv & y_cv:       ',X_cv.shape, y_cv.shape)
print('The shape of X_test & y_test:   ',X_test.shape, y_test.shape)
```

```
The shape of X_train & y_train:  (49041, 17) (49041,)
The shape of X_cv & y_cv:        (24155, 17) (24155,)
The shape of X_test & y_test:    (36052, 17) (36052,)
```

In [16]:

```
# Printing some random text
print(X_train['essay'].values[0])
print("="*127)
print(X_train['essay'].values[500])
print("="*127)
print(X_train['essay'].values[1000])
print("="*127)
print(X_train['essay'].values[10000])
print("="*127)
print(X_train['essay'].values[20000])
print("="*127)
```

My students come from all backgrounds, and are in a high-poverty, rural sc
hool district. They have all different experiences and background knowled
ge.   In my classroom we use technology every day if it is available to u
s. We use computers to help with educating the students in my classroom.
\r\n \r\nWe use computers for individualized instruction in reading and ma
th. We use computers for testing in reading and math. We also use compute
rs for enrichment and remediation in reading and in math. We have differe
nt programs that can help supplement student growth. These programs are a
way to excite the students about their education.\r\n\r\nChromebooks would
open up many learning opportunities for my students. The students will use
the Chromebooks for individualized reading and math instruction through di
fferent websites like Moby Max, Reading Eggs, and Education City, plus oth
ers. This will help each student enrich their learning at their own level
and pace. They will become familiar with the use of a proper keyboard, web
browser, and mouse for the state testing that is required. They will be ab
le to access e-books for digital storytelling and research projects. Googl
e Classroom is another feature that will be able to be used to create diff
erent projects with teacher guidance. Chromebooks will give students acces
s to technology that is not as accessible at this time due to lack of equi
pment availability. \r\n\r\nStudents will be able to move at their own pac
e and be successful. This will help boost confidence and allow learning to
continue. My students will become fluent users of computer technology whic
h will prepare them for computer based state testing.nannan
==========
==========
As a special education teacher in a low-income/high poverty school, my stu
dents are faced with several challenges daily both in and out of the class
room. Despite the many challenges they face, my misson is to create a safe
and warm welcoming classroom while providing my students with meaningful l
essons that allow them to fall in love with learning.\r\n\r\nAll of my stu
dents receive free breakfast and lunch every day. My students are hard wor
king and go above and beyond looking out for one another. It is rewarding
to see how much they grow into leaders every day. \r\n\r\nEvery day, I mak
e it my goal to create a positive learning environment for my students. I
understand that I may not be able to control their home lives, however, I
can certainly control their experience during the school day when they are
with me.\r\nNext year, I will be implementing flexible seating in my class
room. There will be NO desks in my classroom. Instead, the students will h
ave different seating options to choose from such as bean bag chairs, stoo
ls, couches, comfy chairs, benches, etc. \r\n\r\nResearch has shown that g
iving students different seating options increases learning versus regular
desks. \r\n\r\nSince there will not be any desks in my classroom, the stud
ents will need magazine file folders to hold all of their belongings. The
magazine file folders will be stored on a bookshelf in my classroom and th
e students will be able to take them with them to different parts of the r
oom that they are working in.nannan
==========
==========
Dominican Republic, Haiti, Cape Verde, Africa, Latin America. While these
are places many American students read about in books, this is just a samp
ling of areas of the world our students come from. \r\n\r\nImagine walking
into a classroom in the middle of the year knowing minimal English, uncert
ain of what to do or how to communicate. This is what most of my students
deal with after already having left the only people and place they've ever
known. \r\n\r\nMy school is proud to serve a population that is 100% immig
rants, and while most of our students start the year in September, there a
re some who come in January, when everyone else has already hit their stri
de. Although this transition is difficult, as their English teacher, it i
s my duty to welcome them and help them learn as much English as possible
in the minimal time that we have with them.Our students are eager to lear
n, but need the appropriate materials to help them get there. By giving st

udents both fiction and non-fiction books that are low-level English but i nteresting to them, they will feel more engaged in their learning. Some of our students are as old as 21, and to have books that they can read and un derstand, yet also have plots and themes they can connect to, gives them t he confidence to keep learning.\r\n\r\nAs all of my students are English L anguage Learners, sustained reading on a daily basis is imperative for the ir language development. With these books, I will be able to teach them te xts as a whole class or put them in small groups with individualized books and share with each other what they are reading. As many of my students ar e brand new to this country, the literacy they will develop with these boo ks will give them a strong foundation and appreciation for reading that I hope will last a lifetime.nannan

==========

\"For me music is a vehicle to bring our pain to the surface, getting it b ack to that humble and tender spot where, with luck, it can lose its anger and become compassion again.\" - Paula Cole My students escape from their surrounding reality by using music as a tool for self-expression.\r\n\r\nF irst and foremost, my students are compassionate and dedicated.\r\nThey de serve much more then what I currently have for instruments in my music pro gram. Our students come from a very impoverished neighborhood and I have f ound that music has become a great outlet for them. When our students give it their all, their spirits come out, giving us all chills when they perfo rm on stage. Their enthusiasm is endless and they truly follow the mantra of being an explorer. Expectations are set high here for them but they are always exceeded by our students.I started a band at our school just 2 year s ago. What started with just 12 students the first year grew to 50 stude nts last year and this year I am proud to say I can easily enroll 100 band students, both new to music and returning band kids. Now, we just need th e instruments. \r\n\r\nStudents are practically begging for more instrumen ts so we can have more then one band.   Many of our kids look at high scho ol as a means of escaping the poor neighborhood schools surrounding them, and being in this school band gives our kids a leg up on that process. The band program is helping them improve their music reading skills, is prepar ing them for high school band and is also boosting their confidence during their seasonal performances. \r\n\r\nThey have all the passion they can h andle and just need the tools to soar. Now it's time to spread the enjoym ent to more students by providing them with instruments. Thank you for you r interest in this project!\r\nnannan

==========

My students crave technology. They already use it in their daily life at h ome, and now it is time to have technology be a larger part of our classro om! Being able to add more devices to my classroom would enable my student s to take AR quizzes, conduct research projects, use many apps and website s our school utilizes, and so much more!\r\n\r\nWith class sizes always in creasing, student access to tablets or laptops in my classroom has diminis hed.\r\n\r\nBe a part of the solution- help my students access technology daily in their classroom! Thank you for any help you can provide!\r\nIt is so important for all students to have access to technology. There are so m any possibilities for students as they are able to access the many apps an d websites that we can use to enhance their education. \r\nHelp my student s gain access to vital technology that will boost their learning!\r\nIncre ased time for research, typing projects in Google Classroom, and taking AR tests will help each individual student. With this increased time on techn ology, other skills such as basic keyboarding will also improve for each s tudent. Please help me make a positive impact on their education. Thank yo u for anything you can give!nannan

==========

In [17]:

```
# https://stackoverflow.com/a/47091490/4084039
# removing and replacing decontracted phrases.

import re

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

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

In [18]:

```
sent = decontracted(X_train['essay'].values[10000])
print(sent)
print("="*127)
```

```
\"For me music is a vehicle to bring our pain to the surface, getting it b
ack to that humble and tender spot where, with luck, it can lose its anger
and become compassion again.\" - Paula Cole My students escape from their
surrounding reality by using music as a tool for self-expression.\r\n\r\nF
irst and foremost, my students are compassionate and dedicated.\r\nThey de
serve much more then what I currently have for instruments in my music pro
gram. Our students come from a very impoverished neighborhood and I have f
ound that music has become a great outlet for them. When our students give
it their all, their spirits come out, giving us all chills when they perfo
rm on stage. Their enthusiasm is endless and they truly follow the mantra
of being an explorer. Expectations are set high here for them but they are
always exceeded by our students.I started a band at our school just 2 year
s ago.  What started with just 12 students the first year grew to 50 stude
nts last year and this year I am proud to say I can easily enroll 100 band
students, both new to music and returning band kids.  Now, we just need th
e instruments. \r\n\r\nStudents are practically begging for more instrumen
ts so we can have more then one band.   Many of our kids look at high scho
ol as a means of escaping the poor neighborhood schools surrounding them,
and being in this school band gives our kids a leg up on that process. The
band program is helping them improve their music reading skills, is prepar
ing them for high school band and is also boosting their confidence during
their seasonal performances.  \r\n\r\nThey have all the passion they can h
andle and just need the tools to soar.  Now it is time to spread the enjoy
ment to more students by providing them with instruments. Thank you for yo
ur interest in this project!\r\nnannan
===============================================================================
===================================================
```

In [19]:

```
# \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)
```

```
 For me music is a vehicle to bring our pain to the surface, getting it back to that humble and tender spot where, with luck, it can lose its anger and become compassion again.  - Paula Cole My students escape from their surrounding reality by using music as a tool for self-expression.    First and foremost, my students are compassionate and dedicated.  They deserve much more then what I currently have for instruments in my music program. Our students come from a very impoverished neighborhood and I have found that music has become a great outlet for them. When our students give it their all, their spirits come out, giving us all chills when they perform on stage. Their enthusiasm is endless and they truly follow the mantra of being an explorer. Expectations are set high here for them but they are always exceeded by our students.I started a band at our school just 2 years ago.  What started with just 12 students the first year grew to 50 students last year and this year I am proud to say I can easily enroll 100 band students, both new to music and returning band kids.  Now, we just need the instruments.     Students are practically begging for more instruments so we can have more then one band.   Many of our kids look at high school as a means of escaping the poor neighborhood schools surrounding them, and being in this school band gives our kids a leg up on that process. The band program is helping them improve their music reading skills, is preparing them for high school band and is also boosting their confidence during their seasonal performances.      They have all the passion they can handle and just need the tools to soar.  Now it is time to spread the enjoyment to more students by providing them with instruments. Thank you for your interest in this project!  nannan
```

In [20]:

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

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

```
 For me music is a vehicle to bring our pain to the surface getting it bac
k to that humble and tender spot where with luck it can lose its anger and
become compassion again Paula Cole My students escape from their surroundi
ng reality by using music as a tool for self expression First and foremost
my students are compassionate and dedicated They deserve much more then wh
at I currently have for instruments in my music program Our students come
from a very impoverished neighborhood and I have found that music has beco
me a great outlet for them When our students give it their all their spiri
ts come out giving us all chills when they perform on stage Their enthusia
sm is endless and they truly follow the mantra of being an explorer Expect
ations are set high here for them but they are always exceeded by our stud
ents I started a band at our school just 2 years ago What started with jus
t 12 students the first year grew to 50 students last year and this year I
am proud to say I can easily enroll 100 band students both new to music an
d returning band kids Now we just need the instruments Students are practi
cally begging for more instruments so we can have more then one band Many
of our kids look at high school as a means of escaping the poor neighborho
od schools surrounding them and being in this school band gives our kids a
leg up on that process The band program is helping them improve their musi
c reading skills is preparing them for high school band and is also boosti
ng their confidence during their seasonal performances They have all the p
assion they can handle and just need the tools to soar Now it is time to s
pread the enjoyment to more students by providing them with instruments Th
ank you for your interest in this project nannan
```

In [21]:

```
from nltk.corpus import stopwords
stopwords = stopwords.words('english')   # To remove words that comes under the stopwor
d.
print(stopwords)
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "yo
u're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselv
es', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'hersel
f', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'the
mselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'thes
e', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'hav
e', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'th
e', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'a
t', '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', 'on
ce', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'no
t', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can',
'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll',
'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't",
'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't",
'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',
"mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't",
'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "would
n't"]
```

1.3.2 Preprocessing of train data: (X_train['essay'])

In [22]:

```
# Combining all the above statements.

from tqdm import tqdm
preprocessed_essay_train = []

for sentence in tqdm(X_train['essay'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r',' ')
    sent = sent.replace('\\"',' ')
    sent = sent.replace('\\n',' ')
    sent = re.sub('[^A-Za-z0-9]+',' ',sent)
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essay_train.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████
█████| 49041/49041 [00:46<00:00, 1047.45it/s]
```

In [23]:

```
# After preprocessing
preprocessed_essay_train[10000]
```

Out[23]:

```
'for music vehicle bring pain surface getting back humble tender spot luck
lose anger become compassion paula cole my students escape surrounding rea
lity using music tool self expression first foremost students compassionat
e dedicated they deserve much i currently instruments music program our st
udents come impoverished neighborhood i found music become great outlet wh
en students give spirits come giving us chills perform stage their enthusi
asm endless truly follow mantra explorer expectations set high always exce
eded students i started band school 2 years ago what started 12 students f
irst year grew 50 students last year year i proud say i easily enroll 100
band students new music returning band kids now need instruments students
practically begging instruments one band many kids look high school means
escaping poor neighborhood schools surrounding school band gives kids leg
process the band program helping improve music reading skills preparing hi
gh school band also boosting confidence seasonal performances they passion
handle need tools soar now time spread enjoyment students providing instru
ments thank interest project nannan'
```

1.3.3 Preprocessing of cross validation data: (X_cv['essay'])

In [24]:

```
preprocessed_essay_cv = []

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)
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essay_cv.append(sent.lower().strip())
```

100%|██████████| 24155/24155 [00:24<00:00, 974.78it/s]

In [25]:

```
# After preprocessing:
preprocessed_essay_cv[10000]
```

Out[25]:

```
'i teach high poverty rate inner city school approximately 400 students k
5 my fabulous fourth graders consist 13 boys 10 girls range socio economic
backgrounds academic interests abilities there three english language lear
ners two new america there two languages english japanese spanish spoken r
oom my students vary learning styles visual auditory kinesthetic learners
combination students different learning styles need modality accommodation
s our bodies built move little fidgets hands feet squirming hard plastic s
eats learning full time job my students able freely move utilize six stand
desks four zenergy chairs eight wobble stools my goal provide student stan
d desk over next two years projects focus goal benefit students as gears m
ind turn must bodies in 4th grade students expected engage effectively ran
ge collaborative discussions one one groups teacher led diverse partners g
rade 4 topics texts building others ideas expressing clearly when children
sedentary class tend zone fall asleep the engaged invested become educatio
n my fourth graders greatly benefit able move school day uninterrupted way
s the six standing desks previous projects cool moves room cool moves room
ii provide opportunities positive meaningful collaboration around room acr
oss subject areas allowing positive mindful learning community they huge s
uccess i believe adding three standing desks assist students acquiring nec
essary 21st century skills effectively communicate peers across subject ar
eas my students already noticed benefits standing desks they need opportun
ities move often actively seek appropriate input productively interacting
peers demonstrating conceptual knowledge with less restrictions movement a
ttention detail endurance academic rigor improved increased success effect
ively communicating peers positively impacts acquisition knowledge across
subject areas nannan'
```

1.3.4 Preprocessing of test data: (X_test['essay'])

In [26]:

```
preprocessed_essay_test = []

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)
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essay_test.append(sent.lower().strip())
```

100%|██████████| 36052/36052 [00:37<00:00, 960.45it/s]

In [27]:

```
# After preprocessing
preprocessed_essay_test[10000]
```

Out[27]:

```
'the students school special help make family loving caring compassionate
individuals a place people go beyond call duty make sure students things n
eed succeed classroom survive outside our school pre k thru 12th school lo
cated small rural community when students come school want know family hel
p one another anyway if look schools playground would see students standin
g around board our school recently combined form pre k thru 12th grade sch
ool equipment school currently equipment moved prior elementary school nea
rly 8 years ago since nothing changed when asked one thing students would
change improve school stated would like playground equipment the funnel ba
ll set requested would give students something new fun look forward this s
et would used students pre k thru 5th grade this set would also help devel
op strengthen students basketball skills playground basketball goals nanna
n'
```

1.4 Preprocessing of Project titles :

In [28]:

```
# Printing some random project titles:
print(X_train['project_title'].values[0])
print("="*60)
print(X_train['project_title'].values[500])
print("="*60)
print(X_train['project_title'].values[1000])
print("="*60)
print(X_train['project_title'].values[10000])
print("="*60)
print(X_train['project_title'].values[20000])
print("="*60)
```

```
\"Chrometastic\" Learning!
============================================================
Picture This: A Classroom with NO Desks!
============================================================
Help Our Immigrant Students Read English (and Like It!)
============================================================
Woodwinds, Brass, Percussion, OH MY!
============================================================
Time for Tech!
============================================================
```

1.4.1 Preprocessing project title of train data: (X_train['project_title'])

In [29]:

```
preprocessed_title_train = []

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

```
100%|████████████████████████████████████████████████████████████████
████| 49041/49041 [00:02<00:00, 20235.55it/s]
```

In [30]:

```
# After preprocessing:
preprocessed_title_train[20000]
```

Out[30]:

```
'time tech'
```

1.4.2 Preprocessing project title of Cross validation data: (X_cv['project_title'])

In [31]:

```
preprocessed_title_cv = []

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

100%|██| 24155/24155 [00:01<00:00, 20091.34it/s]

In [32]:

```
# After preprocessing :
preprocessed_title_cv[10000]
```

Out[32]:

```
'cool moves room iii'
```

1.4.3 Preprocessing project title of test data: (X_test['project_title'])

In [33]:

```
preprocessed_title_test = []

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

100%|██| 36052/36052 [00:01<00:00, 21416.94it/s]

In [34]:

```
# After preprocessing :
preprocessed_title_test[10000]
```

Out[34]:

```
'funnel ball fun'
```

1.5 Preparing Data for Models :

In [35]:

```
project_data.columns
```

Out[35]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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_approve
d',
       'clean_categories', 'clean_subcategories', 'essay'],
      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_essay - Text data
- project_title - Text data
- project_resource_summary - Text data (optional)

- quantity - numerical (optional)
- 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/)

a. One Hot Encoding of School State :

In [36]:

```
# Using CountVectorizer to convert the categorical data into one hote encoders:
my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())

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

In [37]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_state = CountVectorizer(vocabulary = list(sorted_dict_state_cat.keys()), low
ercase=False, binary=True)
vectorizer_state.fit(X_train['school_state'].values)

school_state_one_hot_train = vectorizer_state.transform(X_train['school_state'].values)
school_state_one_hot_cv = vectorizer_state.transform(X_cv['school_state'].values)
school_state_one_hot_test = vectorizer_state.transform(X_test['school_state'].values)

print(vectorizer_state.get_feature_names())
print("="*90)

print("The shape of school_state_one_hot_train : ",school_state_one_hot_train.shape)
print("The shape of school_state_one_hot_cv    : ",school_state_one_hot_cv.shape)
print("The shape of school_state_one_hot_test  : ",school_state_one_hot_test.shape)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME',
'HI', 'DC', 'NM', 'KS', 'IA', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'N
V', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'M
A', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'N
Y', 'TX', 'CA']
==========================================================================
================
The shape of school_state_one_hot_train :  (49041, 51)
The shape of school_state_one_hot_cv    :  (24155, 51)
The shape of school_state_one_hot_test  :  (36052, 51)
```

b. One Hot Encoding of clean_categories :

In [38]:

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

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

print(vectorizer_cat.get_feature_names())
print("="*90)

print("The shape of categories_one_hot_train: ", categories_one_hot_train.shape)
print("The shape of categories_one_hot_cv   : ", categories_one_hot_cv.shape)
print("The shape of categories_one_hot_test : ", categories_one_hot_test.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
==========================================================================
================
The shape of categories_one_hot_train:  (49041, 9)
The shape of categories_one_hot_cv   :  (24155, 9)
The shape of categories_one_hot_test :  (36052, 9)
```

c. One Hot Encoding of clean_subcategories :

In [39]:

```
vectorizer_subcat = CountVectorizer(vocabulary = list(sorted_sub_cat_dict.keys()), lowe
rcase = False, binary = True)
vectorizer_subcat.fit(X_train['clean_subcategories'].values)

subcategories_one_hot_train = vectorizer_subcat.transform(X_train['clean_subcategories'
].values)
subcategories_one_hot_cv = vectorizer_subcat.transform(X_cv['clean_subcategories'].valu
es)
subcategories_one_hot_test = vectorizer_subcat.transform(X_test['clean_subcategories'].
values)

print(vectorizer_subcat.get_feature_names())
print("="*90)

print("The shape of subcategories_one_hot_train: ", subcategories_one_hot_train.shape)
print("The shape of subcategories_one_hot_cv   : ", subcategories_one_hot_cv.shape)
print("The shape of subcategories_one_hot_test : ", subcategories_one_hot_test.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvemen
t', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'Nutrition
Education', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Musi
c', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'A
ppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Lit
eracy']
==========================================================================
================
The shape of subcategories_one_hot_train:  (49041, 30)
The shape of subcategories_one_hot_cv   :  (24155, 30)
The shape of subcategories_one_hot_test :  (36052, 30)
```

d. One Hot Encoding of project_grade_category :

In [40]:

```
# Setting project_data['project_grade_category'].column :-
project_grade_category = []

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

project_grade_category[0:5]
```

Out[40]:

```
['Grades_PreK-2', 'Grades_6-8', 'Grades_6-8', 'Grades_PreK-2', 'Grades_Pre
K-2']
```

In [41]:

```
project_data['project_grade_category'] = project_grade_category

project_data['project_grade_category'].head(5)
```

Out[41]:

```
55660    Grades_PreK-2
76127       Grades_6-8
51140       Grades_6-8
473      Grades_PreK-2
41558    Grades_PreK-2
Name: project_grade_category, dtype: object
```

In [42]:

```
my_counter = Counter()

for grades in project_data['project_grade_category'].values:
    my_counter.update(grades.split())

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

In [43]:

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

project_grade_category_one_hot_train = vectorizer_grade.transform(X_train['project_grad
e_category'].values)
project_grade_category_one_hot_cv = vectorizer_grade.transform(X_cv['project_grade_cate
gory'].values)
project_grade_category_one_hot_test = vectorizer_grade.transform(X_test['project_grade_
category'].values)

print(vectorizer_grade.get_feature_names())
print("="*80)


print("The shape of project_grade_category_one_hot_train: ", project_grade_category_one
_hot_train.shape)
print("The shape of project_grade_category_one_hot_cv   : ", project_grade_category_one
_hot_cv.shape)
print("The shape of project_grade_category_one_hot_test : ", project_grade_category_one
_hot_test.shape)
```

```
['Grades_9-12', 'Grades_6-8', 'Grades_3-5', 'Grades_PreK-2']
================================================================================
======
The shape of project_grade_category_one_hot_train:  (49041, 4)
The shape of project_grade_category_one_hot_cv   :  (24155, 4)
The shape of project_grade_category_one_hot_test :  (36052, 4)
```

e. One Hot Encoding of teacher_prefix :

In [44]:

```
my_counter = Counter()

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

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

In [45]:

```
vectorizer_prefix = CountVectorizer(vocabulary=list(sorted_teacher_prefix.keys()), lowe
rcase=False, binary=True)
vectorizer_prefix.fit(X_train['teacher_prefix'].values.astype('unicode'))

teacher_prefix_one_hot_train = vectorizer_prefix.transform(X_train['teacher_prefix'].va
lues.astype('unicode'))
teacher_prefix_one_hot_cv = vectorizer_prefix.transform(X_cv['teacher_prefix'].values.a
stype('unicode'))
teacher_prefix_one_hot_test = vectorizer_prefix.transform(X_test['teacher_prefix'].valu
es.astype('unicode'))

print(vectorizer_prefix.get_feature_names())
print("="*80)

print("The shape of teacher_prefix_one_hot_train: ",teacher_prefix_one_hot_train.shape)
print("The shape of teacher_prefix_one_hot_cv   : ",teacher_prefix_one_hot_cv.shape)
print("The shape of teacher_prefix_one_hot_test : ",teacher_prefix_one_hot_test.shape)
```

```
['nan', 'Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
================================================================================
======
The shape of teacher_prefix_one_hot_train:  (49041, 6)
The shape of teacher_prefix_one_hot_cv   :  (24155, 6)
The shape of teacher_prefix_one_hot_test :  (36052, 6)
```

1.5.2 Vectorizing Textual data: (project_essay & project_title)

1.5.2.1 Bag of Words :(project_essay)

a) Bag of word - X_train

In [46]:

```
# We are considering only the words which appeared in at least 10 documents(rows or pro
jects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(preprocessed_essay_train)

project_essay_BOW_train = vectorizer_bow_essay.transform(preprocessed_essay_train)
print("The shape of project_essay_BOW_train after processing: ", project_essay_BOW_trai
n.shape)
```

```
The shape of project_essay_BOW_train after processing:  (49041, 12096)
```

b) Bag of word - X_cv

In [47]:

```
# We are considering only the words which appeared in at least 10 documents(rows or pro
jects).

project_essay_BOW_cv = vectorizer_bow_essay.transform(preprocessed_essay_cv)
print("The shape of project_essay_BOW_cv after processing: ", project_essay_BOW_cv.shap
e)
```

```
The shape of project_essay_BOW_cv after processing:  (24155, 12096)
```

b) Bag of word - X_test

In [48]:

```
# We are considering only the words which appeared in at least 10 documents(rows or pro
jects).

project_essay_BOW_test = vectorizer_bow_essay.transform(preprocessed_essay_test)
print("The shape of project_essay_BOW_test after processing: ", project_essay_BOW_test.
shape)
```

```
The shape of project_essay_BOW_test after processing:  (36052, 12096)
```

Bag of Words :(project_title)

a) X_train

In [49]:

```
vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit(preprocessed_title_train)
project_title_BOW_train = vectorizer_bow_title.transform(preprocessed_title_train)
print("The shape of project_title_BOW_train: ",project_title_BOW_train.shape)
```

```
The shape of project_title_BOW_train:  (49041, 2093)
```

b) X_cv

In [50]:

```
project_title_BOW_cv = vectorizer_bow_title.transform(preprocessed_title_cv)
print("The shape of project_title_cv: ",project_title_BOW_cv.shape)
```

```
The shape of project_title_cv:  (24155, 2093)
```

c) X_test

In [51]:

```
project_title_BOW_test = vectorizer_bow_title.transform(preprocessed_title_test)
print("The shape of project_title_test: ",project_title_BOW_test.shape)
```

```
The shape of project_title_test:  (36052, 2093)
```

1.5.2.2 TFIDF :(project_essay)

a) X_train

In [52]:

```
# We are considering only the words which appeared in at least 10 documents(rows or pro
jects)

from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)

vectorizer_tfidf_essay.fit(preprocessed_essay_train)

project_essay_tfidf_train = vectorizer_tfidf_essay.transform(preprocessed_essay_train)
print("The shape of project_essay_tfidf_train: ", project_essay_tfidf_train.shape)
```

```
The shape of project_essay_tfidf_train:  (49041, 12096)
```

b) X_cv

In [53]:

```
project_essay_tfidf_cv = vectorizer_tfidf_essay.transform(preprocessed_essay_cv)
print("The shape of project_essay_tfidf_cv: ", project_essay_tfidf_cv.shape)
```

```
The shape of project_essay_tfidf_cv:  (24155, 12096)
```

c) X_test

In [54]:

```
project_essay_tfidf_test = vectorizer_tfidf_essay.transform(preprocessed_essay_test)
print("The shape of project_essay_tfidf_test: ", project_essay_tfidf_test.shape)
```

```
The shape of project_essay_tfidf_test:  (36052, 12096)
```

TFIDF :(project_title)

a) X_train

In [55]:

```
vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(preprocessed_title_train)
project_title_tfidf_train = vectorizer_tfidf_title.transform(preprocessed_title_train)

print("The shape of project_title_tfidf_train: ",project_title_tfidf_train.shape)
```

```
The shape of project_title_tfidf_train:  (49041, 2093)
```

b) X_cv

In [56]:

```
project_title_tfidf_cv = vectorizer_tfidf_title.transform(preprocessed_title_cv)

print("The shape of project_title_tfidf_cv: ",project_title_tfidf_cv.shape)
```

```
The shape of project_title_tfidf_cv:  (24155, 2093)
```

c) X_test

In [57]:

```
project_title_tfidf_test = vectorizer_tfidf_title.transform(preprocessed_title_test)

print("The shape of project_title_tfidf_test: ",project_title_tfidf_test.shape)
```

```
The shape of project_title_tfidf_test:  (36052, 2093)
```

1.5.3 Encoding Numerical Features:

a) Price

In [58]:

```
resource_data.head()
```

Out[58]:

	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
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

In [59]:

```
# 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()
```

Out[59]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21
2	p000003	298.97	4
3	p000004	1113.69	98
4	p000005	485.99	8

In [60]:

```
# We need to join the X_train, X_cv and X_test with price_data to proceed further.

X_train = pd.merge(X_train,price_data, on = 'id', how = 'left')
X_cv    = pd.merge(X_cv,   price_data, on = 'id', how = 'left')
X_test  = pd.merge(X_test, price_data, on = 'id', how = 'left')
```

In [61]:

```
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.
```

In [62]:

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

price_train = normalizer.transform(X_train['price'].values.reshape(1,-1)).T
price_cv = normalizer.transform(X_cv['price'].values.reshape(1,-1)).T
price_test = normalizer.transform(X_test['price'].values.reshape(1,-1)).T


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

b) Quantity:

In [63]:

```
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.
```

In [64]:

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

quantity_train = normalizer.transform(X_train['quantity'].values.reshape(1,-1)).T
quantity_cv = normalizer.transform(X_cv['quantity'].values.reshape(1,-1)).T
quantity_test = normalizer.transform(X_test['quantity'].values.reshape(1,-1)).T


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

c) Number of previously posted projects by Teachers:

In [65]:

```
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.
```

In [66]:

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

previously_posted_projects_train = normalizer.transform(X_train['teacher_number_of_prev
iously_posted_projects'].values.reshape(1,-1)).T
previously_posted_projects_cv = normalizer.transform(X_cv['teacher_number_of_previously
_posted_projects'].values.reshape(1,-1)).T
previously_posted_projects_test = normalizer.transform(X_test['teacher_number_of_previo
usly_posted_projects'].values.reshape(1,-1)).T


print("After vectorizations")
print(previously_posted_projects_train.shape, y_train.shape)
print(previously_posted_projects_cv.shape, y_cv.shape)
print(previously_posted_projects_test.shape, y_test.shape)
print("="*100)
```

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

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)**

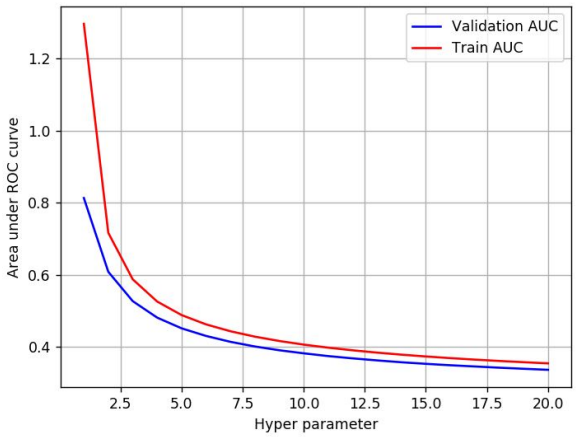
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
    - 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**

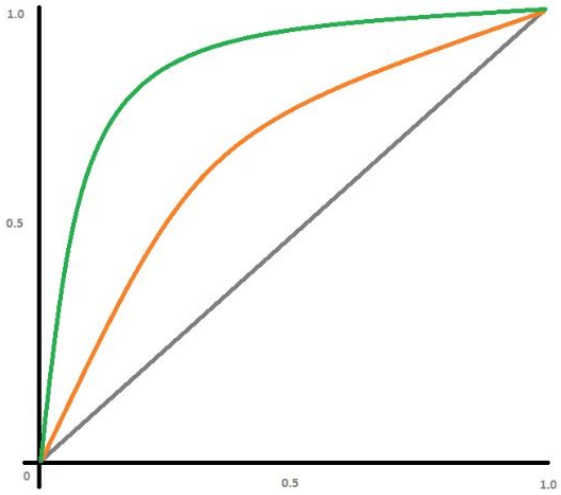
    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using values of `feature_log_prob_` parameter of MultinomialNB (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print their corresponding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

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

(https://seaborn.pydata.org/generated/seaborn.heatmap.html)
(https://seaborn.pydata.org/generated/seaborn.heatmap.html)
(https://seaborn.pydata.org/generated/seaborn.heatmap.html)

(https://seaborn.pydata.org/generated/seaborn.heatmap.html)

5. **Conclusion** (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

(https://seaborn.pydata.org/generated/seaborn.heatmap.html)

- 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 (https://seaborn.pydata.org/generated/seaborn.heatmap.html) link (http://zetcode.com/python/prettytable/)

2. Naive Bayes

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

In [67]:

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

X_tr = hstack((school_state_one_hot_train,categories_one_hot_train,subcategories_one_ho
t_train,
project_grade_category_one_hot_train,teacher_prefix_one_hot_train,price_train,quantity_
train,
previously_posted_projects_train,project_essay_BOW_train,project_title_BOW_train)).tocs
r()

X_cr = hstack((school_state_one_hot_cv,categories_one_hot_cv,subcategories_one_hot_cv,
project_grade_category_one_hot_cv,teacher_prefix_one_hot_cv,price_cv,quantity_cv,
previously_posted_projects_cv,project_essay_BOW_cv,project_title_BOW_cv)).tocsr()

X_te = hstack((school_state_one_hot_test,categories_one_hot_test,subcategories_one_hot_
test,
project_grade_category_one_hot_test,teacher_prefix_one_hot_test,price_test,quantity_tes
t,
previously_posted_projects_test,project_essay_BOW_test,project_title_BOW_test)).tocsr()



print("The 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('='*50)
```

```
The Final Data Matrix :----
(49041, 14292) (49041,)
(24155, 14292) (24155,)
(36052, 14292) (36052,)
==================================================
```

i) Finding the best hyper parameter i.e alpha which results in the maximum AUC value :

GridSearchCV using cv = 10

In [68]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB

MNB = MultinomialNB()
parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1,
0.5, 1, 5,
10, 50, 100, 500, 1000, 2500, 5000, 10000]}
clf = GridSearchCV(MNB, 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']
```

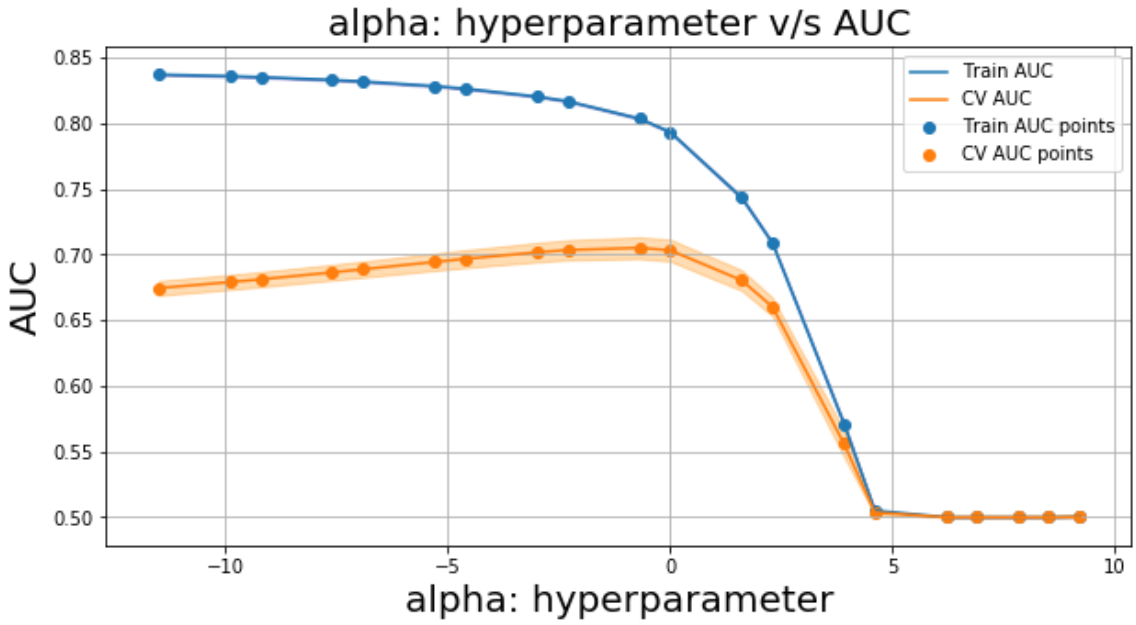
In [69]:

```
# Taking log of all alpha values
import math

alpha = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5,
10, 50, 100, 500, 1000, 2500, 5000, 10000]

log_alpha = []
for values in tqdm(alpha):
    a = math.log(values)
    log_alpha.append(a)
```

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

In [70]:

```
# Plotting AUC vs alpha: hyperparameter curve

plt.figure(figsize=(10,5))
plt.plot(log_alpha, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alpha,train_auc - train_auc_std,train_auc + train_auc_std,al
pha=0.3,color='darkblue')
plt.plot(log_alpha, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alpha,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,colo
r='darkorange')
plt.scatter(log_alpha, train_auc, label='Train AUC points')
plt.scatter(log_alpha, cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha: hyperparameter", fontsize = 20)
plt.ylabel("AUC", fontsize = 20)
plt.title("alpha: hyperparameter v/s AUC", fontsize = 20)
plt.grid()
plt.show()

print("The Best Hyperparater is: ", clf.best_estimator_)
```




```
The Best Hyperparater is:  MultinomialNB(alpha=0.5, class_prior=None, fit_
prior=True)
```

Summary

1. AUC plot of train and cross validation data vs Hyperpater alpha ranging from .00001 to 10000 is shown above.

2. alpha = 0.000001 works very good for the train data as shown in the fig but that value doesnot works well for cross validation data.

3. alpha value when to 1, seems to work well for both train and cross validation data.

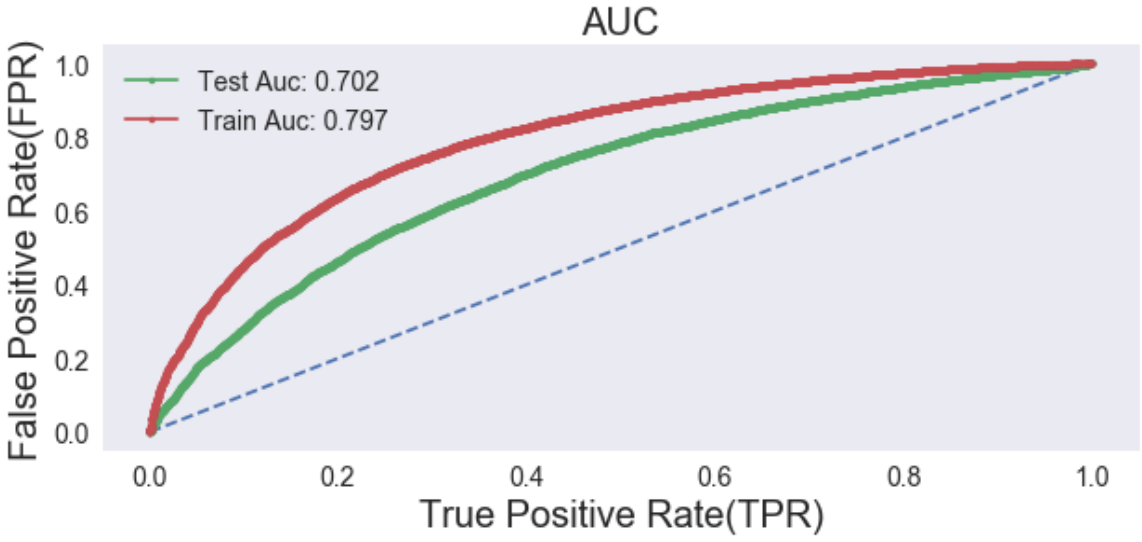
4. alpha values greater than 1 works very bad for the both.

Best value of alpha = 0.5

ii) Training the model using the best alpha value i.e 0.5

In [94]:

```
best_alpha_1 = 0.5

# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc, roc_auc_score
model = MultinomialNB(alpha = 0.5, class_prior=(0.5,0.5))
model.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
# predict probabilities for train dataset.
y_train_prob = model.predict_proba(X_tr)
# keep probabilities for the positive outcome only
y_train_prob = y_train_prob[:, 1]
auc_train = roc_auc_score(y_train, y_train_prob)
train_fpr,train_tpr,train_threshold = roc_curve(y_train, y_train_prob)
# predict probabilities for test dataset.
y_test_prob = model.predict_proba(X_te)
# keep probabilities for the positive outcome only
y_test_prob  = y_test_prob [:, 1]
# calculate AUC
auc = roc_auc_score(y_test, y_test_prob )
#print('AUC: %.3f' % auc)
# calculate roc curve
test_fpr, test_tpr, test_threshold = roc_curve(y_test, y_test_prob )
# plot no skill
plt.figure(figsize=(10,4))
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(test_fpr, test_tpr,label = 'Test Auc: %.3f' % auc, marker='.')
plt.plot(train_fpr, train_tpr,label = 'Train Auc: %.3f' % auc_train, marker='.')
plt.legend()
plt.xlabel("True Positive Rate(TPR)", fontsize = 20)
plt.ylabel("False Positive Rate(FPR)",fontsize = 20)
plt.title("AUC",fontsize = 20)
plt.grid()
# show the plot
plt.show()
```



iii) Confusion Matrix

In [95]:

```
from sklearn.metrics import confusion_matrix
```

In [96]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(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.rou
nd(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [97]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_prob, train_threshold, train_fpr, train
_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_prob, test_threshold, test_fpr, test_tpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5289072794461199 for threshold 0.384
[[ 5234  2192]
 [10595 31020]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4228392877487362 for threshold 0.763
[[ 3602  1857]
 [11128 19465]]
```

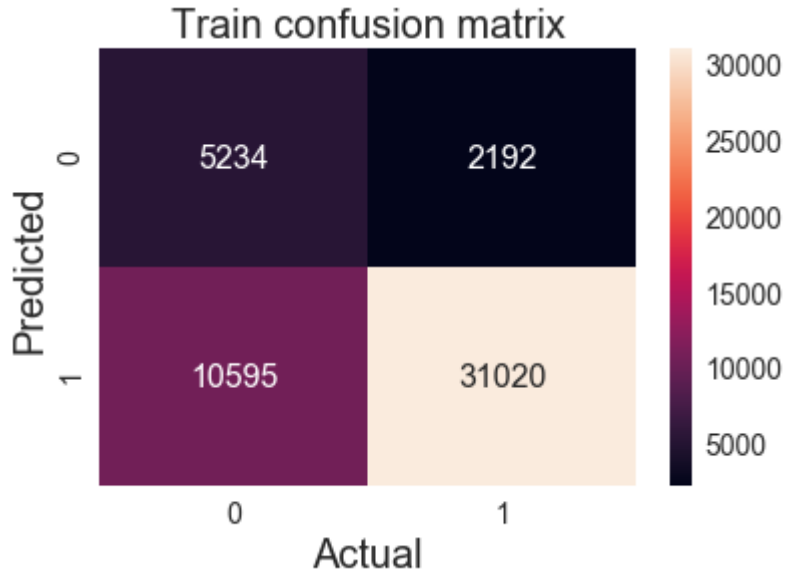
In [98]:

```
CM_df_tr = pd.DataFrame(confusion_matrix(y_train, predict(y_train_prob, train_threshold
,train_fpr, train_tpr)))
```

```
the maximum value of tpr*(1-fpr) 0.5289072794461199 for threshold 0.384
```

In [99]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(CM_df_tr, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Actual', fontsize = 20)
plt.ylabel('Predicted', fontsize = 20)
plt.title('Train confusion matrix', fontsize = 20)
plt.show()
```



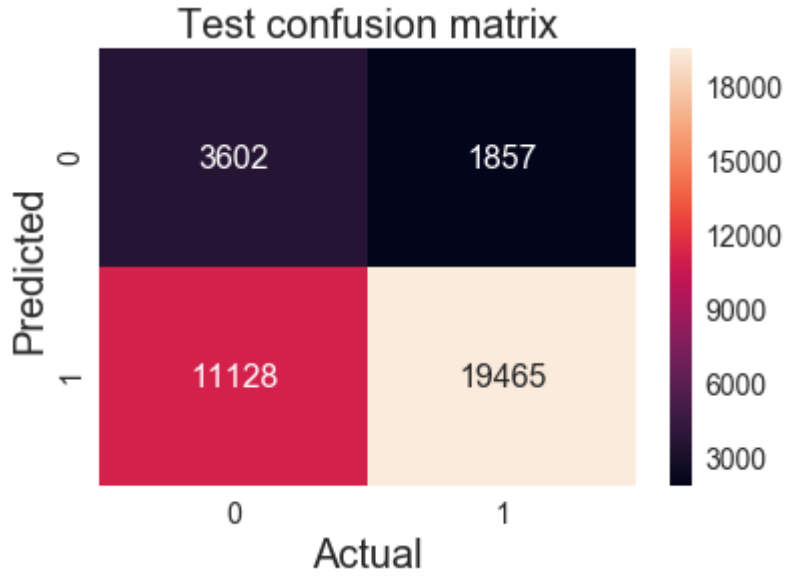
In [100]:

```
CM_df_te = pd.DataFrame(confusion_matrix(y_test, predict(y_test_prob, test_threshold,te
st_fpr, test_tpr)))
```

```
the maximum value of tpr*(1-fpr) 0.4228392877487362 for threshold 0.763
```

In [101]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(CM_df_te, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Actual', fontsize = 20)
plt.ylabel('Predicted', fontsize = 20)
plt.title('Test confusion matrix', fontsize = 20)
plt.show()
```



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

In [102]:

```
X_tr = hstack((school_state_one_hot_train,categories_one_hot_train,subcategories_one_ho
t_train,
project_grade_category_one_hot_train,teacher_prefix_one_hot_train,price_train,quantity_
train,
previously_posted_projects_train,project_essay_tfidf_train,project_title_tfidf_train)).
tocsr()

X_cr = hstack((school_state_one_hot_cv,categories_one_hot_cv,subcategories_one_hot_cv,
project_grade_category_one_hot_cv,teacher_prefix_one_hot_cv,price_cv,quantity_cv,
previously_posted_projects_cv,project_essay_tfidf_cv,project_title_tfidf_cv)).tocsr()

X_te = hstack((school_state_one_hot_test,categories_one_hot_test,subcategories_one_hot_
test,
project_grade_category_one_hot_test,teacher_prefix_one_hot_test,price_test,quantity_tes
t,
previously_posted_projects_test,project_essay_tfidf_test,project_title_tfidf_test)).toc
sr()



print("The 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('='*50)
```

```
The Final Data Matrix :----
(49041, 14292) (49041,)
(24155, 14292) (24155,)
(36052, 14292) (36052,)
==================================================
```

i) Finding the best hyper parameter i.e alpha which results in the maximum AUC value :

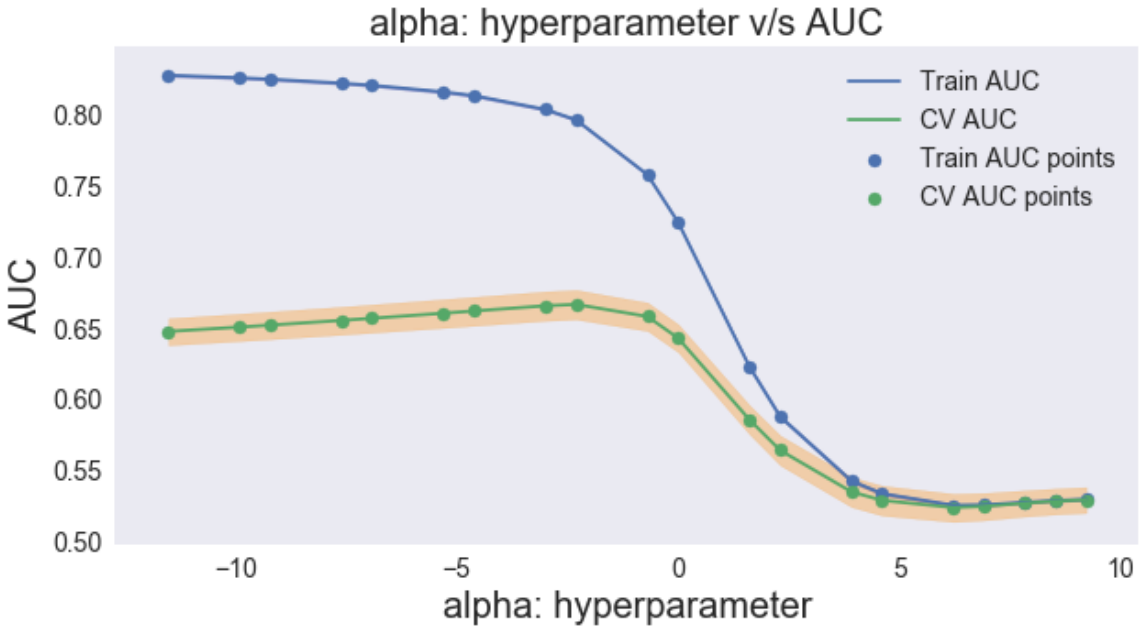
GridSearchCV using cv = 10

In [103]:

```
MNB = MultinomialNB()
parameters = {'alpha':[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1,
0.5, 1, 5,
10, 50, 100, 500, 1000, 2500, 5000, 10000]}
clf = GridSearchCV(MNB, 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 [104]:

```
# Plotting AUC vs alpha: hyperparameter curve

plt.figure(figsize=(10,5))
plt.plot(log_alpha, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alpha,train_auc - train_auc_std,train_auc + train_auc_std,al
pha=0.3,color='darkblue')
plt.plot(log_alpha, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alpha,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,colo
r='darkorange')
plt.scatter(log_alpha, train_auc, label='Train AUC points')
plt.scatter(log_alpha, cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha: hyperparameter", fontsize = 20)
plt.ylabel("AUC", fontsize = 20)
plt.title("alpha: hyperparameter v/s AUC", fontsize = 20)
plt.grid()
plt.show()

print("The Best Hyperparater is: ", clf.best_estimator_)
```




```
The Best Hyperparater is:  MultinomialNB(alpha=0.1, class_prior=None, fit_
prior=True)
```

Summary

1. AUC plot of train and cross validation data vs Hyperpater alpha ranging from .00001 to 10000 is shown above.

2. alpha = 0.000001 works very good for the train data as shown in the fig but that value doesnot works well for cross validation data.

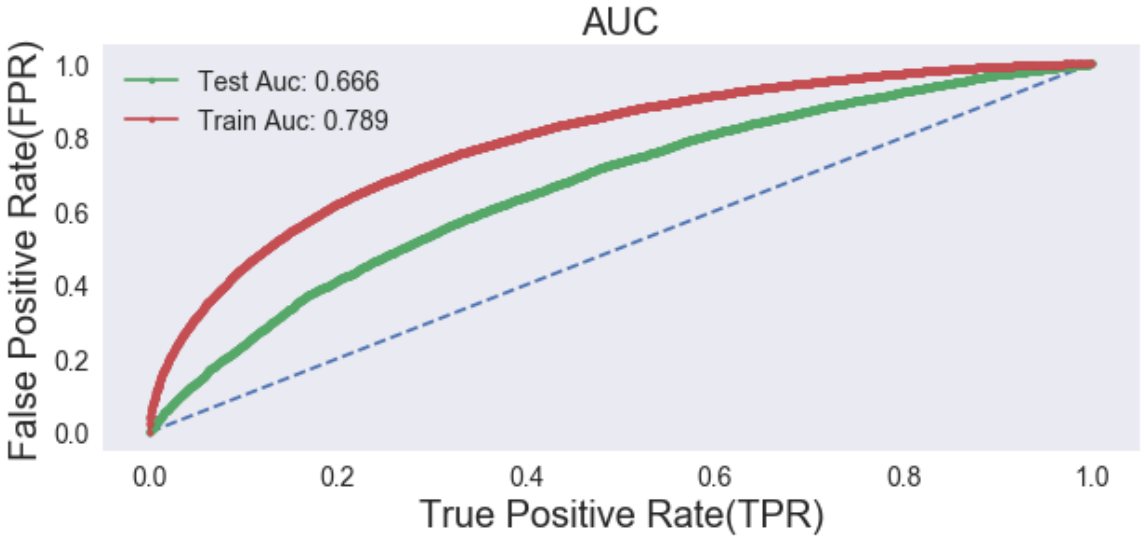
3. alpha value when close to 0.1, seems to work well for both train and cross validation data.

4. alpha values greater than 0.1 works very bad for the both.

ii) Training the model using the best alpha value i.e 0.1

In [105]:

```
best_alpha_1 = 0.1

# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc, roc_auc_score
model = MultinomialNB(alpha = 0.1, class_prior=(0.5,0.5))
model.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
# predict probabilities for train dataset.
y_train_prob = model.predict_proba(X_tr)
# keep probabilities for the positive outcome only
y_train_prob = y_train_prob[:, 1]
auc_train = roc_auc_score(y_train, y_train_prob)
train_fpr,train_tpr,train_threshold = roc_curve(y_train, y_train_prob)
# predict probabilities for test dataset.
y_test_prob = model.predict_proba(X_te)
# keep probabilities for the positive outcome only
y_test_prob  = y_test_prob [:, 1]
# calculate AUC
auc = roc_auc_score(y_test, y_test_prob )
#print('AUC: %.3f' % auc)
# calculate roc curve
test_fpr, test_tpr, test_threshold = roc_curve(y_test, y_test_prob )
# plot no skill
plt.figure(figsize=(10,4))
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
plt.plot(test_fpr, test_tpr,label = 'Test Auc: %.3f' % auc, marker='.')
plt.plot(train_fpr, train_tpr,label = 'Train Auc: %.3f' % auc_train, marker='.')
plt.legend()
plt.xlabel("True Positive Rate(TPR)", fontsize = 20)
plt.ylabel("False Positive Rate(FPR)",fontsize = 20)
plt.title("AUC",fontsize = 20)
plt.grid()
# show the plot
plt.show()
```



iii) Confusion Matrix

In [106]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_prob, train_threshold, train_fpr, train
_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_prob, test_threshold, test_fpr, test_tpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5111635815969033 for threshold 0.505
[[ 5361  2065]
 [12237 29378]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.38635664823215066 for threshold 0.538
[[ 3286  2173]
 [11093 19500]]
```

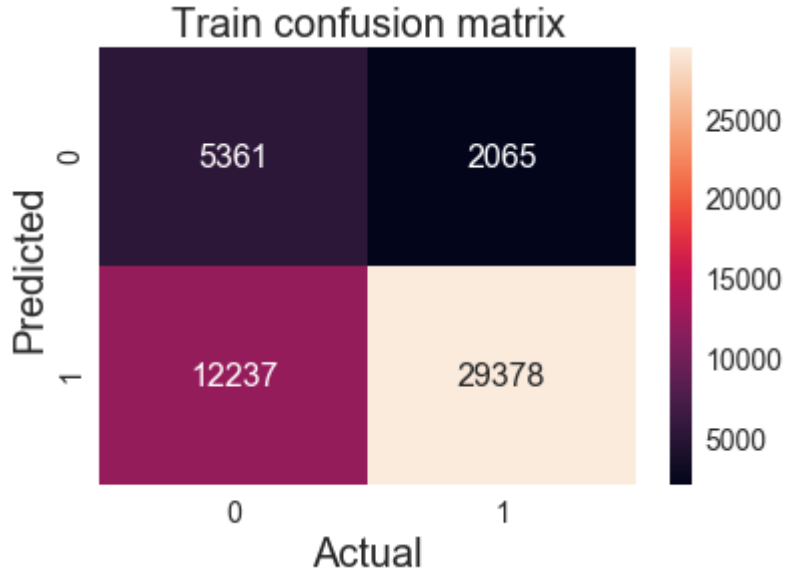
In [107]:

```
CM_df_tr = pd.DataFrame(confusion_matrix(y_train, predict(y_train_prob, train_threshold
,train_fpr, train_tpr)))
```

```
the maximum value of tpr*(1-fpr) 0.5111635815969033 for threshold 0.505
```

In [108]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(CM_df_tr, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Actual', fontsize = 20)
plt.ylabel('Predicted', fontsize = 20)
plt.title('Train confusion matrix', fontsize = 20)
plt.show()
```



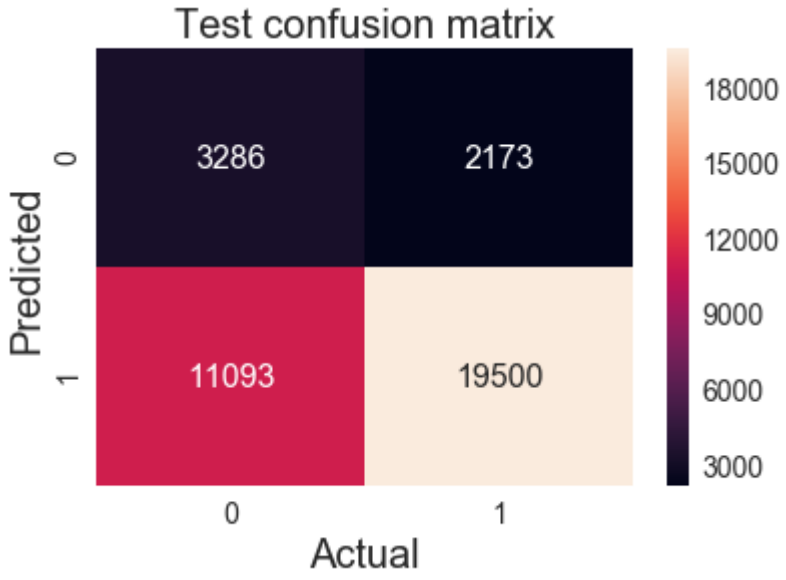
In [109]:

```
CM_df_te = pd.DataFrame(confusion_matrix(y_test, predict(y_test_prob, test_threshold,te
st_fpr, test_tpr)))
```

```
the maximum value of tpr*(1-fpr) 0.38635664823215066 for threshold 0.538
```

In [110]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(CM_df_te, annot=True,annot_kws={"size": 16}, fmt='g')
plt.xlabel('Actual', fontsize = 20)
plt.ylabel('Predicted', fontsize = 20)
plt.title('Test confusion matrix', fontsize = 20)
plt.show()
```



Feature Importance

Finding the top 20 features of positive class and top 20 features of negative class for BOW model (Set 1)

Set 1:

In [111]:

```
X_tr = hstack((school_state_one_hot_train,categories_one_hot_train,subcategories_one_ho
t_train,
project_grade_category_one_hot_train,teacher_prefix_one_hot_train,price_train,quantity_
train,
previously_posted_projects_train,project_essay_BOW_train,project_title_BOW_train)).tocs
r()

X_cr = hstack((school_state_one_hot_cv,categories_one_hot_cv,subcategories_one_hot_cv,
project_grade_category_one_hot_cv,teacher_prefix_one_hot_cv,price_cv,quantity_cv,
previously_posted_projects_cv,project_essay_BOW_cv,project_title_BOW_cv)).tocsr()

X_te = hstack((school_state_one_hot_test,categories_one_hot_test,subcategories_one_hot_
test,
project_grade_category_one_hot_test,teacher_prefix_one_hot_test,price_test,quantity_tes
t,
previously_posted_projects_test,project_essay_BOW_test,project_title_BOW_test)).tocsr()
```

In [112]:

```
MNB = MultinomialNB(alpha = 0.5)
MNB.fit(X_tr, y_train)
```

Out[112]:

```
MultinomialNB(alpha=0.5, class_prior=None, fit_prior=True)
```

In [113]:

```
print("The total number rows & columns in our dataset: ", X_tr.shape)
```

```
The total number rows & columns in our dataset:  (49041, 14292)
```

In [115]:

```
bow_features_probs_negative = []
for a in range(14292) :     # as number of columns in our dataset is 14348 and so is to
tal features
    bow_features_probs_negative.append(MNB.feature_log_prob_[0,a])
```

In [116]:

```
print(len(bow_features_probs_negative))
```

```
14292
```

In [117]:

```
bow_feature_names = []

for a in vectorizer_state.get_feature_names():
    bow_feature_names.append(a)
```

In [118]:

```
for a in vectorizer_cat.get_feature_names():
    bow_feature_names.append(a)
```

In [119]:

```
for a in vectorizer_subcat.get_feature_names():
    bow_feature_names.append(a)
```

In [120]:

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

In [121]:

```
for a in vectorizer_prefix.get_feature_names():
    bow_feature_names.append(a)
```

In [122]:

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

In [123]:

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

In [124]:

```
bow_feature_names.append('price')
bow_feature_names.append('quantity')
bow_feature_names.append('teacher_number_of_previously_posted_projects')
```

In [125]:

```
print(len(bow_feature_names))
```

```
14292
```

In [126]:

```
final_bow_features_negative = pd.DataFrame({'feature_names':bow_feature_names, 'feature
_probs':bow_features_probs_negative})
```

In [127]:

```
df = final_bow_features_negative.sort_values(by = ['feature_probs'], ascending = True)
```

Top 20 Negative features from BOW model (Set 1)

In [128]:

```
df.head(20)
```

Out[128]:

	feature_names	feature_probs
9996	slots	-14.589319
7068	monolingual	-14.589319
3226	digestive	-14.589319
7069	monopoly	-14.589319
7072	monster	-14.589319
12245	after	-14.589319
7083	moon	-14.589319
12260	along	-14.589319
7124	movable	-14.589319
7137	mr	-14.589319
1181	baseballs	-14.589319
1175	barrier	-14.589319
13264	learner	-14.589319
7154	multiplying	-14.589319
2005	checks	-14.589319
7161	mural	-14.589319
1172	baritone	-14.589319
12263	alphabet	-14.589319
2006	cheer	-14.589319
7186	mysteries	-14.589319

In [130]:

```
bow_features_probs_positive = []

for a in range(14292) :     # as number of columns in our dataset is 14348 and so is to
tal features
    bow_features_probs_positive.append(MNB.feature_log_prob_[1,a])
```

In [131]:

```
final_bow_features_positive = pd.DataFrame({'feature_names':bow_feature_names, 'feature
_probs':bow_features_probs_positive})
```

In [132]:

```
df = final_bow_features_positive.sort_values(by = ['feature_probs'], ascending = True)
```

Top 20 Positive features from BOW model (Set 1)

In [133]:

```
df.head(20)
```

Out[133]:

	feature_names	feature_probs
92	Grades_3-5	-16.365288
98	Ms.	-16.365288
97	Mr.	-16.365288
90	Grades_9-12	-16.365288
95	Dr.	-16.365288
91	Grades_6-8	-16.365288
99	Mrs.	-16.365288
93	Grades_PreK-2	-16.365288
94	nan	-15.266676
3802	empower	-14.168063
13414	mobile	-14.168063
14208	wedo	-13.967393
12250	aid	-13.967393
2689	counties	-13.967393
12627	craving	-13.800339
1676	bustling	-13.800339
9480	sample	-13.800339
4766	gap	-13.800339
8862	recognize	-13.800339
11447	underserved	-13.800339

Finding the top 20 features of positive class and top 20 features of negative class for Tfidf model (Set 2)

In [134]:

```
X_tr = hstack((school_state_one_hot_train,categories_one_hot_train,subcategories_one_ho
t_train,
project_grade_category_one_hot_train,teacher_prefix_one_hot_train,price_train,quantity_
train,
previously_posted_projects_train,project_essay_tfidf_train,project_title_tfidf_train)).
tocsr()

X_cr = hstack((school_state_one_hot_cv,categories_one_hot_cv,subcategories_one_hot_cv,
project_grade_category_one_hot_cv,teacher_prefix_one_hot_cv,price_cv,quantity_cv,
previously_posted_projects_cv,project_essay_tfidf_cv,project_title_tfidf_cv)).tocsr()

X_te = hstack((school_state_one_hot_test,categories_one_hot_test,subcategories_one_hot_
test,
project_grade_category_one_hot_test,teacher_prefix_one_hot_test,price_test,quantity_tes
t,
previously_posted_projects_test,project_essay_tfidf_test,project_title_tfidf_test)).toc
sr()
```

In [135]:

```
MNB = MultinomialNB(alpha = 0.1)
MNB.fit(X_tr, y_train)
```

Out[135]:

```
MultinomialNB(alpha=0.1, class_prior=None, fit_prior=True)
```

In [136]:

```
print("The total number rows & columns in our dataset: ", X_tr.shape)
```

```
The total number rows & columns in our dataset:  (49041, 14292)
```

In [137]:

```
tfidf_features_probs_negative = []
for a in range(14292) :     # as number of columns in our dataset is 14348 and so is to
tal features
    tfidf_features_probs_negative.append(MNB.feature_log_prob_[0,a])
```

In [138]:

```
print(len(bow_features_probs_negative))
```

```
14292
```

In [139]:

```
tfidf_feature_names = []

for a in vectorizer_state.get_feature_names():
    tfidf_feature_names.append(a)
```

In [140]:

```
for a in vectorizer_cat.get_feature_names():
    tfidf_feature_names.append(a)
```

In [141]:

```
for a in vectorizer_subcat.get_feature_names():
    tfidf_feature_names.append(a)
```

In [142]:

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

In [143]:

```
for a in vectorizer_prefix.get_feature_names():
    tfidf_feature_names.append(a)
```

In [144]:

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

In [145]:

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

In [146]:

```
tfidf_feature_names.append('price')
tfidf_feature_names.append('quantity')
tfidf_feature_names.append('teacher_number_of_previously_posted_projects')
```

In [147]:

```
final_tfidf_features_negative = pd.DataFrame({'feature_names':tfidf_feature_names, 'fea
ture_probs':tfidf_features_probs_negative})
```

In [148]:

```
df = final_tfidf_features_negative.sort_values(by = ['feature_probs'], ascending = True
)
```

Top 20 Negative features from Tfidf model (Set 2)

In [149]:

```
df.head(20)
```

Out[149]:

	feature_names	feature_probs
728	analysis	-13.903923
11333	tube	-13.903923
12191	zoology	-13.903923
2264	combinations	-13.903923
3968	eraser	-13.903923
13702	re	-13.903923
11326	try	-13.903923
13113	house	-13.903923
12198	101	-13.903923
9984	slight	-13.903923
3979	es	-13.903923
11321	trusted	-13.903923
4996	greatest	-13.903923
4991	gravity	-13.903923
7434	oak	-13.903923
2234	colleagues	-13.903923
6355	legitimate	-13.903923
2228	collaborators	-13.903923
9380	ropes	-13.903923
4034	everlasting	-13.903923

In [150]:

```
tfidf_features_probs_positive = []
for a in range(14292) :     # as number of columns in our dataset is 14348 and so is to
tal features
    tfidf_features_probs_positive.append(MNB.feature_log_prob_[1,a])
```

In [151]:

```
final_tfidf_features_positive = pd.DataFrame({'feature_names':tfidf_feature_names, 'fea
ture_probs':tfidf_features_probs_positive})
```

In [152]:

```
df = final_tfidf_features_positive.sort_values(by = ['feature_probs'], ascending = True
)
```

Top 20 Positive features from Tfidf model (Set 2)

In [153]:

```
df.head(20)
```

Out[153]:

	feature_names	feature_probs
95	Dr.	-15.629613
97	Mr.	-15.629613
98	Ms.	-15.629613
99	Mrs.	-15.629613
92	Grades_3-5	-15.629613
93	Grades_PreK-2	-15.629613
90	Grades_9-12	-15.629613
91	Grades_6-8	-15.629613
2689	counties	-13.650278
3802	empower	-13.542445
9072	repairing	-13.431431
9485	sanctuary	-13.422379
4482	flexibility	-13.393303
7449	observational	-13.390330
9357	rogers	-13.385192
5179	he	-13.372995
8862	recognize	-13.367576
2271	come	-13.365052
7035	modification	-13.363044
3502	drastically	-13.357512

3. Conclusions

In [154]:

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

# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prett
ytable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Alpha:Hyper Parameter", "Test_AUC"]
x.add_row(["BOW", "Naive Bayes", 0.5, 0.702])
x.add_row(["TFIDF", "Naive Bayes", 0.1, 0.66])

print(x)
```

```
+------------+-------------+-----------------------+----------+
| Vectorizer |    Model    | Alpha:Hyper Parameter | Test_AUC |
+------------+-------------+-----------------------+----------+
|    BOW     | Naive Bayes |          0.5          |  0.702   |
|   TFIDF    | Naive Bayes |          0.1          |   0.66   |
+------------+-------------+-----------------------+----------+
```

Comparing with KNN

In [155]:

```
y = PrettyTable()
y.field_names = ["Vectorizer", "Model", "Hyper Parameter", "Test_AUC"]
y.add_row(["BOW", "KNN-Brute", 93, 0.66])
y.add_row(["TFIDF", "KNN-Brute", 85, 0.57])
print(y)
```

```
+------------+-----------+-----------------+----------+
| Vectorizer |   Model   | Hyper Parameter | Test_AUC |
+------------+-----------+-----------------+----------+
|    BOW     | KNN-Brute |        93       |   0.66   |
|   TFIDF    | KNN-Brute |        85       |   0.57   |
+------------+-----------+-----------------+----------+
```

Summary

1. Comparing the table we can say the Naive Bayes performs better than KNN for both BOW & Tfidf

2. Naive Bayes is much more faster than KNN.