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:** `p03`
`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`
`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](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_co) **Example**
`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 mar 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

Feature	Descri
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. **Exa** 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 **Exampl**

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [0]:

```
%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
```

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

1.1 Reading Data

In [0]:

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

In [0]:

```
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' 'sc
hool_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 [0]:

```
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[0]:

	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 [0]:

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

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

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

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 [0]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling 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/2289859
5/4084039
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 [0]:

```
# 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 [0]:

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

Out[0]:

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

In [0]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [0]:

```
# 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)
```

```
My students are English learners that are working on English as their second
or third languages. We are a melting pot of refugees, immigrants, and native-
born Americans bringing the gift of language to our school. \r\n\r\n We have
over 24 languages represented in our English Learner program with students at
every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and ex
periences to us that open our eyes to new cultures, beliefs, and respect.\"Th
e limits of your language are the limits of your world.\"-Ludwig Wittgenstein
Our English learner's have a strong support system at home that begs for more
resources.  Many times our parents are learning to read and speak English alo
ng side of their children.  Sometimes this creates barriers for parents to be
able to help their child learn phonetics, letter recognition, and other readi
ng skills.\r\n\r\nBy providing these dvd's and players, students are able to
continue their mastery of the English language even if no one at home is able
to assist.  All families with students within the Level 1 proficiency status,
will be a offered to be a part of this program.  These educational videos wil
l be specially chosen by the English Learner Teacher and will be sent home re
gularly to watch.  The videos are to help the child develop early reading ski
lls.\r\n\r\nParents that do not have access to a dvd player will have the opp
ortunity to check out a dvd player to use for the year.  The plan is to use t
hese videos and educational dvd's for the years to come for other EL student
s.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year al
l love learning, at least most of the time. At our school, 97.3% of the stude
nts receive free or reduced price lunch. Of the 560 students, 97.3% are minor
ity students. \r\nThe school has a vibrant community that loves to get togeth
er and celebrate. Around Halloween there is a whole school parade to show off
the beautiful costumes that students wear. On Cinco de Mayo we put on a big f
estival with crafts made by the students, dances, and games. At the end of th
e year the school hosts a carnival to celebrate the hard work put in during t
he school year, with a dunk tank being the most popular activity.My students
will use these five brightly colored Hokki stools in place of regular, statio
nary, 4-legged chairs. As I will only have a total of ten in the classroom an
d not enough for each student to have an individual one, they will be used in
a variety of ways. During independent reading time they will be used as speci
al chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of th
e day they will be used by the students who need the highest amount of moveme
nt in their life in order to stay focused on school.\r\n\r\nWhenever asked wh
at the classroom is missing, my students always say more Hokki Stools. They c
an't get their fill of the 5 stools we already have. When the students are si
tting in group with me on the Hokki Stools, they are always moving, but at th
e same time doing their work. Anytime the students get to pick where they can
sit, the Hokki Stools are the first to be taken. There are always students wh
o head over to the kidney table to get one of the stools who are disappointed
as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for
7 hours a day. The Hokki stools will be a compromise that allow my students t
o do desk work and move at the same time. These stools will help students to
meet their 60 minutes a day of movement by allowing them to activate their co
re muscles for balance while they sit. For many of my students, these chairs
will take away the barrier that exists in schools for a child who can't sit s
till.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with
plain walls, rows of desks, and a teacher in front of the room? A typical day
in our room is nothing like that. I work hard to create a warm inviting theme
```

```
d room for my students look forward to coming to each day.\r\n\r\nMy class is
made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r
\nThey attend a Title I school, which means there is a high enough percentage
of free and reduced-price lunch to qualify. Our school is an \"open classroom
\" concept, which is very unique as there are no walls separating the classro
oms. These 9 and 10 year-old students are very eager learners; they are like
sponges, absorbing all the information and experiences and keep on wanting mo
re.With these resources such as the comfy red throw pillows and the whimsical
nautical hanging decor and the blue fish nets, I will be able to help create
the mood in our classroom setting to be one of a themed nautical environment.
Creating a classroom environment is very important in the success in each and
every child's education. The nautical photo props will be used with each chil
d as they step foot into our classroom for the first time on Meet the Teacher
evening. I'll take pictures of each child with them, have them developed, and
then hung in our classroom ready for their first day of 4th grade.  This kind
gesture will set the tone before even the first day of school! The nautical t
hank you cards will be used throughout the year by the students as they creat
e thank you cards to their team groups.\r\n\r\nYour generous donations will h
elp me to help make our classroom a fun, inviting, learning environment from
day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to g
et our classroom ready. Please consider helping with this project to make our
new school year a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations. \r\n\r\nThe materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or red
uced price lunch.  Despite their disabilities and limitations, my students lo
ve coming to school and come eager to learn and explore.Have you ever felt li
ke you had ants in your pants and you needed to groove and move as you were i
n a meeting? This is how my kids feel all the time. The want to be able to mo
ve as they learn or so they say.Wobble chairs are the answer and I love then
because they develop their core, which enhances gross motor and in Turn fine
motor skills. \r\nThey also want to learn through games, my kids don't want t
o sit and do worksheets. They want to learn to count by jumping and playing.
Physical engagement is the key to our success. The number toss and color and
shape mats can make that happen. My students will forget they are doing work
and just have the fun a 6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher d
emonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school ha
s 803 students which is makeup is 97.6% African-American, making up the large
st segment of the student body. A typical school in Dallas is made up of 23.
2% African-American students. Most of the students are on free or reduced lun
ch. We aren't receiving doctors, lawyers, or engineers children from rich bac
kgrounds or neighborhoods. As an educator I am inspiring minds of young child
ren and we focus not only on academics but one smart, effective, efficient, a
nd disciplined students with good character.In our classroom we can utilize t
he Bluetooth for swift transitions during class. I use a speaker which does
n't amplify the sound enough to receive the message. Due to the volume of my
speaker my students can't hear videos or books clearly and it isn't making th
e lessons as meaningful. But with the bluetooth speaker my students will be a
ble to hear and I can stop, pause and replay it at any time.\r\nThe cart will
allow me to have more room for storage of things that are needed for the day
and has an extra part to it I can use.  The table top chart has all of the le
tter, words and pictures for students to learn about different letters and it
```

```
is more accessible.nannan
==================================================
```

In [0]:

```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 [0]:

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

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations. \r\n\r\nThe materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or red
uced price lunch.  Despite their disabilities and limitations, my students lo
ve coming to school and come eager to learn and explore.Have you ever felt li
ke you had ants in your pants and you needed to groove and move as you were i
n a meeting? This is how my kids feel all the time. The want to be able to mo
ve as they learn or so they say.Wobble chairs are the answer and I love then
because they develop their core, which enhances gross motor and in Turn fine
motor skills. \r\nThey also want to learn through games, my kids do not want
to sit and do worksheets. They want to learn to count by jumping and playing.
Physical engagement is the key to our success. The number toss and color and
shape mats can make that happen. My students will forget they are doing work
and just have the fun a 6 year old deserves.nannan
==================================================
```

In [0]:

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

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations.     The materials we have are the ones I seek out for my students. I
teach in a Title I school where most of the students receive free or reduced
price lunch.  Despite their disabilities and limitations, my students love co
ming to school and come eager to learn and explore.Have you ever felt like yo
u had ants in your pants and you needed to groove and move as you were in a m
eeting? This is how my kids feel all the time. The want to be able to move as
they learn or so they say.Wobble chairs are the answer and I love then becaus
e they develop their core, which enhances gross motor and in Turn fine motor
skills.   They also want to learn through games, my kids do not want to sit a
nd do worksheets. They want to learn to count by jumping and playing. Physica
l engagement is the key to our success. The number toss and color and shape m
ats can make that happen. My students will forget they are doing work and jus
t have the fun a 6 year old deserves.nannan
```

In [0]:

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

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays cognitive delays gross fine motor delays to autism They are eage
r beavers and always strive to work their hardest working past their limitati
ons The materials we have are the ones I seek out for my students I teach in
a Title I school where most of the students receive free or reduced price lun
ch Despite their disabilities and limitations my students love coming to scho
ol and come eager to learn and explore Have you ever felt like you had ants i
n your pants and you needed to groove and move as you were in a meeting This
is how my kids feel all the time The want to be able to move as they learn or
so they say Wobble chairs are the answer and I love then because they develop
their core which enhances gross motor and in Turn fine motor skills They also
want to learn through games my kids do not want to sit and do worksheets They
want to learn to count by jumping and playing Physical engagement is the key
to our success The number toss and color and shape mats can make that happen
My students will forget they are doing work and just have the fun a 6 year ol
d deserves nannan
```

In [0]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'
, "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he'
, 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'it
self', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becau
se', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'a
ll', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d'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", 'm
a', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [0]:

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

```
100%|████████████████████████████████████████████████████████████████████████
█| 109248/109248 [02:14<00:00, 814.43it/s]
```

In [0]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[0]:
```
'my kindergarten students varied disabilities ranging speech language delays
cognitive delays gross fine motor delays autism they eager beavers always str
ive work hardest working past limitations the materials ones i seek students
i teach title i school students receive free reduced price lunch despite disa
bilities limitations students love coming school come eager learn explore hav
e ever felt like ants pants needed groove move meeting this kids feel time th
e want able move learn say wobble chairs answer i love develop core enhances
gross motor turn fine motor skills they also want learn games kids not want s
it worksheets they want learn count jumping playing physical engagement key s
uccess the number toss color shape mats make happen my students forget work f
un 6 year old deserves nannan'
```

1.4 Preprocessing of `project_title`

In [0]:
```
# similarly you can preprocess the titles also
```

1.5 Preparing data for models

In [0]:
```
project_data.columns
```

Out[0]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', '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_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 [0]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (109248, 9)
```

In [0]:
```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowe
rcase=False, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcateg
ories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducati
on', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geo
graphy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'Env
ironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'Spec
ialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

In [0]:
```
# you can do the similar thing with state, teacher_prefix and project_grade_ca
tegory also
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [0]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

In [0]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

1.5.2.2 TFIDF vectorizer

In [0]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [0]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/408403
9
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# ============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coup
us", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how
-to-use-pickle-to-save-and-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)
```

Out[0]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/40
84039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n
f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in t
qdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n
embedding = np.array([float(val) for val in splitLine[1:]])\n        model[wo
rd] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return
model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ==================
==========\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/
s]\nDone. 1917495  words loaded!\n\n# ============================\n\nwords =
[]\nfor i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in p
reproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in t
he coupus", len(words))\nwords = set(words)\nprint("the unique words in the c
oupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\np
rint("The number of words that are present in both glove vectors and our coup
us",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,
3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in wor
ds:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("wo
rd 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle
files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-v
ariables-in-python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as
f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [0]:

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

In [0]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████
| 109248/109248 [01:06<00:00, 1631.10it/s]

109248
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [0]:

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

In [0]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in th
is list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████████
█| 109248/109248 [08:30<00:00, 214.10it/s]

109248
300
```

In [0]:

```
# Similarly you can vectorize for title also
```

1.5.3 Vectorizing Numerical features

In [0]:

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

In [0]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.    ... 399.    287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

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

In [0]:

```
price_standardized
```

Out[0]:

```
array([[0.00098843, 0.00191166, 0.00330448, ..., 0.00153418, 0.00046704,
        0.00070265]])
```

1.5.4 Merging all the above features

- we need to merge all the numerical vectors i.e catogorical, text, numerical vectors

In [0]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

In [0]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

Out[0]:

```
(109248, 16663)
```

In [0]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Computing Sentiment Scores

In [0]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = '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 wide range\
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 o
ur school is a caring community of successful \
learners which can be seen through collaborative student project based learnin
g in and out of the classroom kindergarteners \
in my class love to work with hands on materials and have many different oppor
tunities to practice a skill before it is\
mastered having the social skills to work cooperatively with friends is a cruc
ial aspect of the kindergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my stude
nts love to role play in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try coo
king with real food i will take their idea \
and create common core cooking lessons where we learn important math and writi
ng concepts while cooking delicious healthy \
food for snack time my students will have a grounded 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 bodi
es this project would expand our learning of \
nutrition and agricultural cooking recipes 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 enjoyment for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
D:\installed\Anaconda3\lib\site-packages\nltk\twitter\__init__.py:20: UserWar
ning:

The twython library has not been installed. Some functionality from the twitt
er package will not be available.


neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

Assignment 9: RF and GBDT

Response Coding: Example

> The response tabel is built only on train dataset. For a category which is not there in train data and present in test data, we will encode them with default values Ex: in our test data if have State: D then we encode it as [0.5, 0.05]

1. **Apply both Random Forrest and GBDT on these feature sets**

    - Set 1: categorical(instead of one hot encoding, try [response coding (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/): use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical(instead of one hot encoding, try [response coding (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/): use probability values), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
    - Set 3: categorical(instead of one hot encoding, try [response coding (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/): use probability values), numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: categorical(instead of one hot encoding, try [response coding (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/): use probability values), numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

2. **The hyper paramter tuning (Consider any two hyper parameters preferably n_estimators, max_depth)**

    - 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/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - find the best hyper paramter using k-fold cross validation/simple cross validation data
    - use gridsearch cv or randomsearch cv or you can write your own for loops to do this task

3. **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

      with X-axis as **n_estimators**, Y-axis as **max_depth**, and Z-axis as **AUC Score** , we have given the notebook which explains how to plot this 3d plot, you can find it in the same drive *3d_scatter_plot.ipynb*

    or

    - 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

      [seaborn heat maps (https://seaborn.pydata.org/generated/seaborn.heatmap.html)](https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**
    - You can choose either of the plotting techniques: 3d plot or heat map
    - 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

4. **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/)

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2. Random Forest and GBDT

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_notebook as tqdm1
from tqdm import tqdm
import time
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

from sklearn.model_selection import train_test_split
```

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' 'sc
hool_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']
```

Text preprocessing(1)

In [4]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

In [5]:

```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(5)
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

In [6]:

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

Out[6]:

```
Counter({'AppliedLearning': 12135,
         'Care_Hunger': 1388,
         'Health_Sports': 14223,
         'History_Civics': 5914,
         'Literacy_Language': 52239,
         'Math_Science': 41421,
         'Music_Arts': 10293,
         'SpecialNeeds': 13642,
         'Warmth': 1388})
```

In [7]:

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))


# ind = np.arange(len(sorted_cat_dict))
# plt.figure(figsize=(20,5))
# p1 = plt.bar(ind, list(sorted_cat_dict.values()))

# plt.ylabel('Projects')
# plt.title('% of projects aproved category wise')
# plt.xticks(ind, list(sorted_cat_dict.keys()))
# plt.show()
# print(sorted_cat_dict)
```

In [8]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

In [9]:
```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[9]:

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

In [10]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

In [11]:
```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))


# ind = np.arange(len(sorted_sub_cat_dict))
# plt.figure(figsize=(20,5))
# p1 = plt.bar(ind, list(sorted_sub_cat_dict.values()))

# plt.ylabel('Projects')
# plt.title('% of projects aproved state wise')
# plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
# plt.show()
```

In [12]:
```
# 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 [13]:

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

Out[13]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [14]:

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

In [15]:

```
#presence of the numerical digits in a strings with numeric : https://stackove
rflow.com/a/19859308/8089731
def hasNumbers(inputString):
    return any(i.isdigit() for i in inputString)
p1 = project_data[['id','project_resource_summary']]
p1 = pd.DataFrame(data=p1)
p1.columns = ['id','digits_in_summary']
p1['digits_in_summary'] =  p1['digits_in_summary'].map(hasNumbers)
# https://stackoverflow.com/a/17383325/8089731
p1['digits_in_summary'] = p1['digits_in_summary'].astype(int)
project_data = pd.merge(project_data, p1, on='id', how='left')
project_data.head(5)
```

Out[15]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

5 rows × 21 columns

Text preprocessing(2)

In [16]:

```
# 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]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'
, "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he'
, 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'it
self', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becau
se', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'a
ll', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d'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", 'm
a', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [18]:

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

```
100%|██████████| 109248/109248 [01:10<00:00, 1550.76it/s]
```

In [19]:

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

```
100%|██████████| 109248/109248 [00:03<00:00, 35330.03it/s]
```

In [20]:

```
preprocessed_titles[1000]
```

Out[20]:

```
'sailing into super 4th grade year'
```

In [21]:

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

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

project_grade_cat_list = []
for i in tqdm1(project_grade_catogories):
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    project_grade_cat_list.append(temp.strip())
```

In [22]:

```
project_data['clean_project_grade_category'] = project_grade_cat_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data.head(2)
```

Out[22]:

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

2 rows × 21 columns

In [23]:
```
project_data.drop(['project_essay_1','project_essay_2','project_essay_3','proj
ect_essay_4'], axis=1, inplace=True)
project_data.head(2)
```

Out[23]:

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

In [24]:
```
#Replacing Nan's with maximum occured value: https://stackoverflow.com/a/51053
916/8089731
project_data['teacher_prefix'].value_counts().argmax()
project_data.fillna(value=project_data['teacher_prefix'].value_counts().argmax
(),axis=1,inplace=True)
```

In [25]:
```
project_data['preprocessed_essays'] = preprocessed_essays
project_data['preprocessed_titles'] = preprocessed_titles
```

In [26]:
```
project_data.columns
```

Out[26]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantit
y',
       'digits_in_summary', 'clean_project_grade_category',
       'preprocessed_essays', 'preprocessed_titles'],
      dtype='object')
```

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

In [27]:
```
X_train, X_test, y_train, y_test = train_test_split(project_data,project_data[
'project_is_approved'], test_size=0.33, stratify = project_data['project_is_ap
proved'])
# X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=
0.33, stratify=y_train)

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

```
(73196, 19)
(36052, 19)
```

In [28]:
```
# please write all the code with proper documentation, and proper titles for e
ach subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debug
ging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the rea
der
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

2.2 Make Data Model Ready: encoding numerical, categorical features

Response coding for categorical data

1.Categories :

In [29]:
```
X_train_pos = X_train.loc[X_train['project_is_approved'] == 1]
```

In [30]:
```
X_train_neg = X_train.loc[X_train['project_is_approved'] == 0]
```

In [31]:
```
clean_cat_pos = {}
for a in X_train_pos['clean_categories'] :
    for b in a.split():
        if b not in clean_cat_pos :
            clean_cat_pos[b] = 1
        else :
            clean_cat_pos[b] += 1
```

In [32]:

```
clean_cat_neg = {}
for a in X_train_neg['clean_categories'] :
    for b in a.split():
        if b not in clean_cat_neg :
            clean_cat_neg[b] = 1
        else :
            clean_cat_neg[b] += 1
```

In [33]:

```
clean_cat_total = {}
for a in X_train['clean_categories'] :
    for b in a.split():
        if b not in clean_cat_total :
            clean_cat_total[b] = 1
        else :
            clean_cat_total[b] += 1
```

In [34]:

```
pos_prob_cat = {}
for pp in clean_cat_total.keys():
    pos_prob_cat[pp] = (clean_cat_pos[pp])/float(clean_cat_total[pp])
```

In [35]:

```
neg_prob_cat = {}
for npp in clean_cat_total.keys():
    neg_prob_cat[npp] = (clean_cat_neg[npp])/float(clean_cat_total[npp])
```

In [36]:

```
cat_0_train = []
cat_1_train = []
for a in X_train["clean_categories"] :
    b = a.split()
    if len(b) == 1 :
        cat_0_train.append(neg_prob_cat[a])
        cat_1_train.append(pos_prob_cat[a])
    else :
        c = neg_prob_cat[b[0]]
        d = neg_prob_cat[b[1]]
        e = pos_prob_cat[b[0]]
        f = pos_prob_cat[b[1]]
        cat_0_train.append(c*d)
        cat_1_train.append(e*f)
```

In [37]:

```
X_train["cat_0"] = cat_0_train
```

In [38]:

```
X_train["cat_1"] = cat_1_train
```

In [39]:

```
cat_0_test = []
cat_1_test = []
for a in X_test["clean_categories"] :
    b = a.split()
    if len(b) == 1 :
        cat_0_test.append(neg_prob_cat[a])
        cat_1_test.append(pos_prob_cat[a])
    else :
        c = neg_prob_cat[b[0]]
        d = neg_prob_cat[b[1]]
        e = pos_prob_cat[b[0]]
        f = pos_prob_cat[b[1]]
        cat_0_test.append(c*d)
        cat_1_test.append(e*f)
```

In [40]:

```
X_test["cat_0"] = cat_0_test
```

In [41]:

```
X_test["cat_1"] = cat_1_test
```

In [42]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.    ... 399.    287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

cat_scalar0 = StandardScaler()
cat_scalar0.fit(X_train['cat_0'].values.reshape(-1,1)) # finding the mean and
 standard deviation of this data

# Now standardize the data with above maen and variance.
cat_0_train = cat_scalar0.transform(X_train['cat_0'].values.reshape(-1, 1))
cat_0_test = cat_scalar0.transform(X_test['cat_0'].values.reshape(-1, 1))
print(cat_0_train.shape)
print(cat_0_test.shape)
```

```
(73196, 1)
(36052, 1)
```

In [43]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

cat_scalar1 = StandardScaler()
cat_scalar1.fit(X_train['cat_1'].values.reshape(-1,1)) # finding the mean and
 standard deviation of this data

# Now standardize the data with above maen and variance.
cat_1_train = cat_scalar1.transform(X_train['cat_1'].values.reshape(-1, 1))
cat_1_test = cat_scalar1.transform(X_test['cat_1'].values.reshape(-1, 1))
print(cat_1_train.shape)
print(cat_1_test.shape)
```

```
(73196, 1)
(36052, 1)
```

2. Sub-Categories :

In [44]:

```
clean_subcat_pos = {}
for a in X_train_pos['clean_subcategories'] :
    for b in a.split():
        if b not in clean_subcat_pos :
            clean_subcat_pos[b] = 1
        else :
            clean_subcat_pos[b] += 1
```

In [45]:

```
clean_subcat_neg = {}
for a in X_train_neg['clean_subcategories'] :
    for b in a.split():
        if b not in clean_subcat_neg :
            clean_subcat_neg[b] = 1
        else :
            clean_subcat_neg[b] += 1
```

In [46]:

```
clean_subcat_total = {}
for a in X_train['clean_subcategories'] :
    for b in a.split():
        if b not in clean_subcat_total :
            clean_subcat_total[b] = 1
        else :
            clean_subcat_total[b] += 1
```

In [47]:

```
pos_prob_subcat = {}
for pp in clean_subcat_total.keys():
    pos_prob_subcat[pp] = (clean_subcat_pos[pp])/float(clean_subcat_total[pp])
```

In [48]:

```
neg_prob_subcat = {}
for npp in clean_subcat_total.keys():
    neg_prob_subcat[npp] = (clean_subcat_neg[npp])/float(clean_subcat_total[np
p])
```

In [49]:

```
subcat_0_train = []
subcat_1_train = []
for a in X_train["clean_subcategories"] :
    b = a.split()
    if len(b) == 1 :
        subcat_0_train.append(neg_prob_subcat[a])
        subcat_1_train.append(pos_prob_subcat[a])
    else :
        c = neg_prob_subcat[b[0]]
        d = neg_prob_subcat[b[1]]
        e = pos_prob_subcat[b[0]]
        f = pos_prob_subcat[b[1]]
        subcat_0_train.append(c*d)
        subcat_1_train.append(e*f)
```

In [50]:

```
X_train["subcat_0"] = subcat_0_train
```

In [51]:

```
X_train["subcat_1"] = subcat_1_train
```

In [52]:

```
subcat_0_test = []
subcat_1_test = []
for a in X_test["clean_subcategories"] :
    b = a.split()
    if len(b) == 1 :
        subcat_0_test.append(neg_prob_subcat[a])
        subcat_1_test.append(pos_prob_subcat[a])
    else :
        c = neg_prob_subcat[b[0]]
        d = neg_prob_subcat[b[1]]
        e = pos_prob_subcat[b[0]]
        f = pos_prob_subcat[b[1]]
        subcat_0_test.append(c*d)
        subcat_1_test.append(e*f)
```

In [53]:

```
X_test["subcat_0"] = subcat_0_test
```

In [54]:

```
X_test["subcat_1"] = subcat_1_test
```

In [55]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

subcat_scalar0 = StandardScaler()
subcat_scalar0.fit(X_train['subcat_0'].values.reshape(-1,1)) # finding the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
subcat_0_train = subcat_scalar0.transform(X_train['subcat_0'].values.reshape(-1, 1))
subcat_0_test = subcat_scalar0.transform(X_test['subcat_0'].values.reshape(-1, 1))
print(subcat_0_train.shape)
print(subcat_0_test.shape)
```

```
(73196, 1)
(36052, 1)
```

In [56]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

subcat_scalar1 = StandardScaler()
subcat_scalar1.fit(X_train['subcat_1'].values.reshape(-1,1)) # finding the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
subcat_1_train = subcat_scalar1.transform(X_train['subcat_1'].values.reshape(-1, 1))
subcat_1_test = subcat_scalar1.transform(X_test['subcat_1'].values.reshape(-1, 1))
print(subcat_1_train.shape)
print(subcat_1_test.shape)
```

```
(73196, 1)
(36052, 1)
```

3. School State :

In [57]:

```
school_state_pos = {}
for a in X_train_pos['school_state'] :
    if a not in school_state_pos :
        school_state_pos[a] = 1
    else :
        school_state_pos[a] += 1
```

In [58]:

```
school_state_neg = {}
for a in X_train_neg['school_state'] :
    if a not in school_state_neg :
        school_state_neg[a] = 1
    else :
        school_state_neg[a] += 1
```

In [59]:

```
school_state_total = {}
for a in X_train['school_state'] :
    if a not in school_state_total :
        school_state_total[a] = 1
    else :
        school_state_total[a] += 1
```

In [60]:

```
pos_prob_state = {}
for state in school_state_total.keys():
    pos_prob_state[state] = (school_state_pos[state])/float(school_state_total
[state])
```

In [61]:

```
neg_prob_state = {}
for state in school_state_total.keys():
    neg_prob_state[state] = (school_state_neg[state])/float(school_state_total
[state])
```

In [62]:

```
state_0_train = []
state_1_train = []
for a in X_train["school_state"] :
    state_0_train.append(neg_prob_state[a])
    state_1_train.append(pos_prob_state[a])
```

In [63]:

```
X_train["state_0"] =state_0_train
X_train["state_1"] = state_1_train
```

In [64]:

```
state_0_test = []
state_1_test = []
for a in X_test["school_state"] :
    state_0_test.append(neg_prob_state[a])
    state_1_test.append(pos_prob_state[a])
```

In [65]:

```
X_test["state_0"] =state_0_test
```

In [66]:

```
X_test["state_1"] =state_1_test
```

In [67]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.    ... 399.    287.73    5.5 ].
# Reshape your data either using array.reshape(-1, 1)

state_scalar0 = StandardScaler()
state_scalar0.fit(X_train['state_0'].values.reshape(-1,1)) # finding the mean
 and standard deviation of this data

# Now standardize the data with above maen and variance.
state_0_train = state_scalar0.transform(X_train['state_0'].values.reshape(-1,
1))
state_0_test = state_scalar0.transform(X_test['state_0'].values.reshape(-1, 1
))
print(state_0_train.shape)
print(state_0_test.shape)
```

```
(73196, 1)
(36052, 1)
```

In [68]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.    ... 399.    287.73    5.5 ].
# Reshape your data either using array.reshape(-1, 1)

state_scalar1 = StandardScaler()
state_scalar1.fit(X_train['state_1'].values.reshape(-1,1)) # finding the mean
 and standard deviation of this data

# Now standardize the data with above maen and variance.
state_1_train = state_scalar1.transform(X_train['state_1'].values.reshape(-1,
1))
state_1_test = state_scalar1.transform(X_test['state_1'].values.reshape(-1, 1
))
print(state_1_train.shape)
print(state_1_test.shape)
```

```
(73196, 1)
(36052, 1)
```

4. Teacher Prefix :

In [69]:

```
teacher_pref_pos = {}
for a in X_train_pos['teacher_prefix'] :
    if a not in teacher_pref_pos :
        teacher_pref_pos[a] = 1
    else :
        teacher_pref_pos[a] += 1
```

In [70]:

```
teacher_pref_neg = {}
for a in X_train_neg['teacher_prefix'] :
    if a not in teacher_pref_neg :
        teacher_pref_neg[a] = 1
    else :
        teacher_pref_neg[a] += 1
```

In [71]:

```
teacher_pref_total = {}
for a in X_train['teacher_prefix'] :
    if a not in teacher_pref_total :
        teacher_pref_total[a] = 1
    else :
        teacher_pref_total[a] += 1
```

In [72]:

```
pos_prob_teacher_pref = {}
for pp in teacher_pref_total.keys():
    pos_prob_teacher_pref[pp] = (teacher_pref_pos[pp])/float(teacher_pref_total[pp])
```

In [73]:

```
neg_prob_teacher_pref = {}
for npp in teacher_pref_total.keys():
    neg_prob_teacher_pref[npp] = (teacher_pref_neg[npp])/float(teacher_pref_total[npp])
```

In [74]:

```
teacher_pref_0_train = []
teacher_pref_1_train = []
for a in X_train["teacher_prefix"] :
    teacher_pref_0_train.append(neg_prob_teacher_pref[a])
    teacher_pref_1_train.append(pos_prob_teacher_pref[a])
```

In [75]:

```
X_train["teacher_prefix_0"] = teacher_pref_0_train
```

In [76]:

```
X_train["teacher_prefix_1"] = teacher_pref_1_train
```

In [77]:

```
teacher_pref_0_test = []
teacher_pref_1_test = []
for a in X_test["teacher_prefix"] :
    teacher_pref_0_test.append(neg_prob_teacher_pref[a])
    teacher_pref_1_test.append(pos_prob_teacher_pref[a])
```

In [78]:

```
X_test["teacher_prefix_0"] = teacher_pref_0_test
```

In [79]:

```
X_test["teacher_prefix_1"] = teacher_pref_1_test
```

In [80]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

teacherpre_scalar0 = StandardScaler()
teacherpre_scalar0.fit(X_train['teacher_prefix_0'].values.reshape(-1,1)) # fin
ding the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
teacher_prefix_0_train = teacherpre_scalar0.transform(X_train['teacher_prefix_
0'].values.reshape(-1, 1))
teacher_prefix_0_test = teacherpre_scalar0.transform(X_test['teacher_prefix_0'
].values.reshape(-1, 1))
print(teacher_prefix_0_train.shape)
print(teacher_prefix_0_test.shape)
```

```
(73196, 1)
(36052, 1)
```

In [81]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

teacherpre_scalar1 = StandardScaler()
teacherpre_scalar1.fit(X_train['teacher_prefix_1'].values.reshape(-1,1)) # fin
ding the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
teacher_prefix_1_train = teacherpre_scalar1.transform(X_train['teacher_prefix_
1'].values.reshape(-1, 1))
teacher_prefix_1_test = teacherpre_scalar1.transform(X_test['teacher_prefix_1'
].values.reshape(-1, 1))
print(teacher_prefix_1_train.shape)
print(teacher_prefix_1_test.shape)
```

```
(73196, 1)
(36052, 1)
```

5. Project Grade :

In [82]:

```
proj_grade_pos = {}
for a in X_train_pos['clean_project_grade_category'] :
    if a not in proj_grade_pos :
        proj_grade_pos[a] = 1
    else :
        proj_grade_pos[a] += 1
```

In [83]:

```
proj_grade_neg = {}
for a in X_train_neg['clean_project_grade_category'] :
    if a not in proj_grade_neg :
        proj_grade_neg[a] = 1
    else :
        proj_grade_neg[a] += 1
```

In [84]:

```
proj_grade_total = {}
for a in X_train['clean_project_grade_category'] :
    if a not in proj_grade_total :
        proj_grade_total[a] = 1
    else :
        proj_grade_total[a] += 1
```

In [85]:

```
pos_prob_grade_cat = {}
for pp in proj_grade_total.keys():
    pos_prob_grade_cat[pp] = (proj_grade_pos[pp])/float(proj_grade_total[pp])
```

In [86]:

```
neg_prob_grade_cat = {}
for npp in proj_grade_total.keys():
    neg_prob_grade_cat[npp] = (proj_grade_neg[npp])/float(proj_grade_total[npp
])
```

In [87]:

```
proj_grade_0_train = []
proj_grade_1_train = []
for a in X_train["clean_project_grade_category"] :
    proj_grade_0_train.append(neg_prob_grade_cat[a])
    proj_grade_1_train.append(pos_prob_grade_cat[a])
```

In [88]:

```
X_train["proj_grade_0"] = proj_grade_0_train
```

In [89]:

```
X_train["proj_grade_1"] = proj_grade_1_train
```

In [90]:

```
proj_grade_0_test = []
proj_grade_1_test = []
for a in X_test["clean_project_grade_category"] :
    proj_grade_0_test.append(neg_prob_grade_cat[a])
    proj_grade_1_test.append(pos_prob_grade_cat[a])
```

In [91]:

```
X_test["proj_grade_0"] = proj_grade_0_test
```

In [92]:

```
X_test["proj_grade_1"] = proj_grade_1_test
```

In [93]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

projectgrade_scalar0 = StandardScaler()
projectgrade_scalar0.fit(X_train['proj_grade_0'].values.reshape(-1,1)) # findi
ng the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
proj_grade_0_train = projectgrade_scalar0.transform(X_train['proj_grade_0'].va
lues.reshape(-1, 1))
proj_grade_0_test = projectgrade_scalar0.transform(X_test['proj_grade_0'].valu
es.reshape(-1, 1))
print(proj_grade_0_train.shape)
print(proj_grade_0_test.shape)
```

```
(73196, 1)
(36052, 1)
```

In [94]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

projectgrade_scalar1 = StandardScaler()
projectgrade_scalar1.fit(X_train['proj_grade_1'].values.reshape(-1,1)) # findi
ng the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
proj_grade_1_train = projectgrade_scalar1.transform(X_train['proj_grade_1'].va
lues.reshape(-1, 1))
proj_grade_1_test = projectgrade_scalar1.transform(X_test['proj_grade_1'].valu
es.reshape(-1, 1))
print(proj_grade_1_train.shape)
print(proj_grade_1_train.shape)
```

```
(73196, 1)
(73196, 1)
```

Vectorizing Numerical features

In [95]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
price_standardized_train = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
# price_standardized_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_standardized_test = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
print(price_standardized_train.shape)
# print(price_standardized_cv.shape)
print(price_standardized_test.shape)
```

```
(73196, 1)
(36052, 1)
```

In [96]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
quantity_standardized_train = quantity_scalar.transform(X_train['quantity'].va
lues.reshape(-1, 1))
# quantity_standardized_cv = quantity_scalar.transform(X_cv['quantity'].value
s.reshape(-1, 1))
quantity_standardized_test = quantity_scalar.transform(X_test['quantity'].valu
es.reshape(-1, 1))
print(quantity_standardized_train.shape)
# print(quantity_standardized_cv.shape)
print(quantity_standardized_test.shape)
```

```
(73196, 1)
(36052, 1)
```

In [97]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 32
9.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized_train = teacher_numb
er_of_previously_posted_projects_scalar.transform(X_train['teacher_number_of_p
reviously_posted_projects'].values.reshape(-1, 1))
# teacher_number_of_previously_posted_projects_standardized_cv = teacher_numbe
r_of_previously_posted_projects_scalar.transform(X_cv['teacher_number_of_previ
ously_posted_projects'].values.reshape(-1, 1))
teacher_number_of_previously_posted_projects_standardized_test = teacher_numbe
r_of_previously_posted_projects_scalar.transform(X_test['teacher_number_of_pre
viously_posted_projects'].values.reshape(-1, 1))
print(teacher_number_of_previously_posted_projects_standardized_train.shape)
# print(teacher_number_of_previously_posted_projects_standardized_cv.shape)
print(teacher_number_of_previously_posted_projects_standardized_test.shape)
```

```
(73196, 1)
(36052, 1)
```

In [98]:

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

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

2.3 Make Data Model Ready: encoding eassay, and project_title

In [99]:
```
X_train.head(2)
```

Out[99]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	proje
72330	147168	p170872	6af4328f5bafd72b9c9c497d3abd2410	Ms.	IL	
10398	65301	p172004	f5b4f781689f214c212ba9ec699780e1	Mrs.	CA	

2 rows × 29 columns

Bag of Words(BOW) on `project_TEXT/ESSAYS` (Train,Cv,Test)

In [100]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws or projects).
vectorizer_bow_essays = CountVectorizer(min_df=10,max_features=5000,ngram_rang
e=(1,2))
vectorizer_bow_essays.fit(X_train['preprocessed_essays'])

text_bow_train = vectorizer_bow_essays.transform(X_train['preprocessed_essays'
])
# text_bow_cv = vectorizer_bow_essays.transform(X_cv['preprocessed_essays'])
text_bow_test = vectorizer_bow_essays.transform(X_test['preprocessed_essays'])
print("Shape of matrix after BOW_text_train ",text_bow_train.shape)
# print("Shape of matrix after BOW_text_cv ",text_bow_cv.shape)
print("Shape of matrix after BOW_text_test ",text_bow_test.shape)
```

```
Shape of matrix after BOW_text_train  (73196, 5000)
Shape of matrix after BOW_text_test  (36052, 5000)
```

Bag of Words(BOW) on `project_title` (Train,Cv,Test)

In [101]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_titles = CountVectorizer(min_df=10)
vectorizer_bow_titles.fit(X_train['preprocessed_titles'])

title_bow_train = vectorizer_bow_titles.transform(X_train['preprocessed_titles'])
# title_bow_cv = vectorizer_bow_titles.transform(X_cv['preprocessed_titles'])
title_bow_test = vectorizer_bow_titles.transform(X_test['preprocessed_titles'])
print("Shape of matrix after BOW_title_train ",title_bow_train.shape)
# print("Shape of matrix after BOW_title_cv ",title_bow_cv.shape)
print("Shape of matrix after BOW_title_test ",title_bow_test.shape)
```

```
Shape of matrix after BOW_title_train  (73196, 2627)
Shape of matrix after BOW_title_test  (36052, 2627)
```

TFIDF Vectorizer on `project_TEXT/ESSAYS` (Train,Cv,Test)

In [ ]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essays = TfidfVectorizer(min_df=10,max_features=5000,ngram_range=(1,2))
vectorizer_tfidf_essays.fit(X_train['preprocessed_essays'])

text_tfidf_train = vectorizer_tfidf_essays.transform(X_train['preprocessed_essays'])
# text_tfidf_cv = vectorizer_tfidf_essays.transform(X_cv['preprocessed_essays'])
text_tfidf_test = vectorizer_tfidf_essays.transform(X_test['preprocessed_essays'])
print("Shape of matrix after tfidf_text_train ",text_tfidf_train.shape)
# print("Shape of matrix after tfidf_text_cv ",text_tfidf_cv.shape)
print("Shape of matrix after tfidf_text_test ",text_tfidf_test.shape)
```

TFIDF Vectorizer on `project_title` (Train,Cv,Test)

In [103]:

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

title_tfidf_train = vectorizer_tfidf_title.transform(X_train['preprocessed_tit
les'])
# title_tfidf_cv = vectorizer_tfidf_title.transform(X_cv['preprocessed_title
s'])
title_tfidf_test = vectorizer_tfidf_title.transform(X_test['preprocessed_title
s'])
print("Shape of matrix after tfidf_title_train ",title_tfidf_train.shape)
# print("Shape of matrix after tfidf_title_cv ",title_tfidf_cv.shape)
print("Shape of matrix after tfidf_title_test ",title_tfidf_test.shape)
```

```
Shape of matrix after tfidf_title_train  (73196, 2638)
Shape of matrix after tfidf_title_test  (36052, 2638)
```

In [104]:

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

Avg W2V on TEXT/ESSAYS(Train,cv,test)

In [105]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_essays_vectors_train = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm1(X_train['preprocessed_essays']): # for each review/sente
nce
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_essays_vectors_train.append(vector)

# avg_w2v_essays_vectors_cv = []; # the avg-w2v for each sentence/review is st
ored in this list
# for sentence in tqdm1(X_cv['preprocessed_essays']): # for each review/senten
ce
#     vector = np.zeros(300) # as word vectors are of zero length
#     cnt_words =0; # num of words with a valid vector in the sentence/review
#     for word in sentence.split(): # for each word in a review/sentence
#         if word in glove_words:
#             vector += model[word]
#             cnt_words += 1
#     if cnt_words != 0:
#         vector /= cnt_words
#     avg_w2v_essays_vectors_cv.append(vector)

avg_w2v_essays_vectors_test = []; # the avg-w2v for each sentence/review is st
ored in this list
for sentence in tqdm1(X_test['preprocessed_essays']): # for each review/senten
ce
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_essays_vectors_test.append(vector)

print(len(avg_w2v_essays_vectors_train))
# print(len(avg_w2v_essays_vectors_cv))
print(len(avg_w2v_essays_vectors_test))
print(len(avg_w2v_essays_vectors_train[0]))
# print(len(avg_w2v_essays_vectors_cv[0]))
print(len(avg_w2v_essays_vectors_test[0]))
```

```
73196
36052
300
300
```

Avg W2V on TITLES(Train,cv,test)

In [106]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_titles_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm1(X_train['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_titles_vectors_train.append(vector)

# avg_w2v_titles_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
# for sentence in tqdm1(X_cv['preprocessed_titles']): # for each review/sentence
#     vector = np.zeros(300) # as word vectors are of zero length
#     cnt_words =0; # num of words with a valid vector in the sentence/review
#     for word in sentence.split(): # for each word in a review/sentence
#         if word in glove_words:
#             vector += model[word]
#             cnt_words += 1
#     if cnt_words != 0:
#         vector /= cnt_words
#     avg_w2v_titles_vectors_cv.append(vector)

avg_w2v_titles_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm1(X_test['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_titles_vectors_test.append(vector)

print(len(avg_w2v_titles_vectors_train))
# print(len(avg_w2v_titles_vectors_cv))
print(len(avg_w2v_titles_vectors_test))
print(len(avg_w2v_titles_vectors_train[0]))
# print(len(avg_w2v_titles_vectors_cv[0]))
print(len(avg_w2v_titles_vectors_test[0]))
```

```
73196
36052
300
300
```

TFIDF weighted W2V on TEXT/ESSAYS(Train,cv,test)

In [107]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_essays'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())
#*****************************************************************************
*
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_essays_vectors_train = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm1(X_train['preprocessed_essays']): # for each review/sente
nce
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_essays_vectors_train.append(vector)


# # average Word2Vec
# # compute average word2vec for each review.
# tfidf_w2v_essays_vectors_cv = []; # the avg-w2v for each sentence/review is
 stored in this list
# for sentence in tqdm1(X_cv['preprocessed_essays']): # for each review/senten
ce
#     vector = np.zeros(300) # as word vectors are of zero length
#     tf_idf_weight =0; # num of words with a valid vector in the sentence/rev
iew
#     for word in sentence.split(): # for each word in a review/sentence
#         if (word in glove_words) and (word in tfidf_words):
#             vec = model[word] # getting the vector for each word
#             # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
#             tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
#             vector += (vec * tf_idf) # calculating tfidf weighted w2v
#             tf_idf_weight += tf_idf
#     if tf_idf_weight != 0:
#         vector /= tf_idf_weight
#     tfidf_w2v_essays_vectors_cv.append(vector)


# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_essays_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm1(X_test['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_essays_vectors_test.append(vector)

print(len(tfidf_w2v_essays_vectors_train))
# print(len(tfidf_w2v_essays_vectors_cv))
print(len(tfidf_w2v_essays_vectors_test))
print(len(tfidf_w2v_essays_vectors_train[0]))
# print(len(tfidf_w2v_essays_vectors_cv[0]))
print(len(tfidf_w2v_essays_vectors_test[0]))
```

```
73196
36052
300
300
```

TFIDF weighted W2V on TITLES(Train,cv,test)

In [108]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_titles'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())
#*****************************************************************************
*
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_titles_vectors_train = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm1(X_train['preprocessed_titles']): # for each review/sente
nce
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_titles_vectors_train.append(vector)


# # average Word2Vec
# # compute average word2vec for each review.
# tfidf_w2v_titles_vectors_cv = []; # the avg-w2v for each sentence/review is
 stored in this list
# for sentence in tqdm1(X_cv['preprocessed_titles']): # for each review/senten
ce
#     vector = np.zeros(300) # as word vectors are of zero length
#     tf_idf_weight =0; # num of words with a valid vector in the sentence/rev
iew
#     for word in sentence.split(): # for each word in a review/sentence
#         if (word in glove_words) and (word in tfidf_words):
#             vec = model[word] # getting the vector for each word
#             # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
#             tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
#             vector += (vec * tf_idf) # calculating tfidf weighted w2v
#             tf_idf_weight += tf_idf
#     if tf_idf_weight != 0:
#         vector /= tf_idf_weight
#     tfidf_w2v_titles_vectors_cv.append(vector)


# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_titles_vectors_test = []; # the avg-w2v for each sentence/review is
 stored in this list
for sentence in tqdm1(X_test['preprocessed_titles']): # for each review/senten
ce
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_titles_vectors_test.append(vector)

print(len(tfidf_w2v_titles_vectors_train))
# print(len(tfidf_w2v_titles_vectors_cv))
print(len(tfidf_w2v_titles_vectors_test))
print(len(tfidf_w2v_titles_vectors_train[0]))
# print(len(tfidf_w2v_titles_vectors_cv[0]))
print(len(tfidf_w2v_titles_vectors_test[0]))
```

```
73196
36052
300
300
```

In [1]:

```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

In [2]:

```
project_data.columns
```

Out[2]:

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

In [ ]:

In [3]:

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

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

2.4 Applying Random Forest

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

2.4.1 Applying Random Forests on BOW, SET 1

In [4]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((cat_0_train, cat_1_train, subcat_0_train, subcat_1_train, state
_0_train, state_1_train
               ,teacher_prefix_0_train,teacher_prefix_1_train, proj_grade_0_tr
ain, proj_grade_1_train
               ,price_standardized_train,quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,text_bow_train,title_bow_train)).tocsr()
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_
one_hot_cv,teacher_prefix_one_hot_cv
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quan
tity_standardized_cv
#                ,teacher_number_of_previously_posted_projects_standardized_c
v,text_bow_cv,title_bow_cv)).tocsr()
X_te = hstack((cat_0_test, cat_1_test, subcat_0_test, subcat_1_test, state_0_t
est, state_1_test
               ,teacher_prefix_0_test,teacher_prefix_1_test,proj_grade_0_test,
proj_grade_1_test
               ,price_standardized_test,quantity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,text_bow_test,title_bow_test )).tocsr()

print("Final Data matrix on BOW")
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 on BOW
(73196, 7652) (73196,)
(36052, 7652) (36052,)
=============================================================================
=======================
```

1.1 Method 1: GridSearchCV

In [ ]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import time


start_time = time.time()
rf = RandomForestClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[10, 50, 100,
500, 1000]}
clf = GridSearchCV(rf, parameters, cv= 3, scoring='roc_auc',return_train_score
=True)
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']
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

In [1]:

```
import dill
# dill.dump_session('notebook_env1.db')
dill.load_session('notebook_env1.db')
```

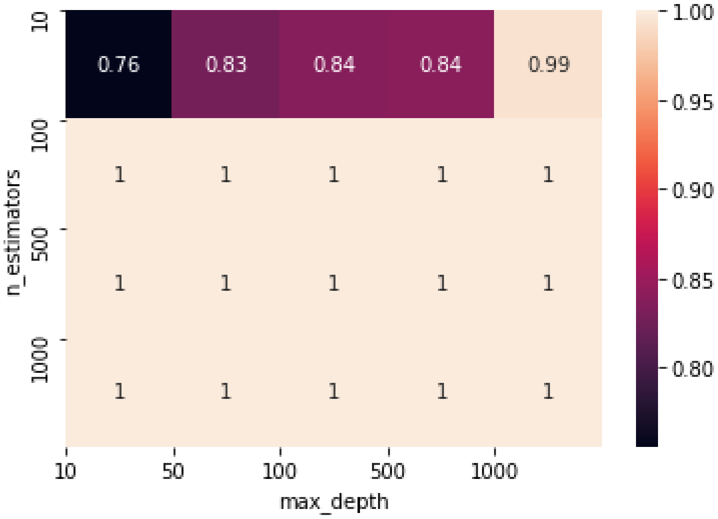
In [2]:

```
train_auc = train_auc.reshape(4,5)
cv_auc = cv_auc.reshape(4,5)
train_auc
cv_auc
```

Out[2]:

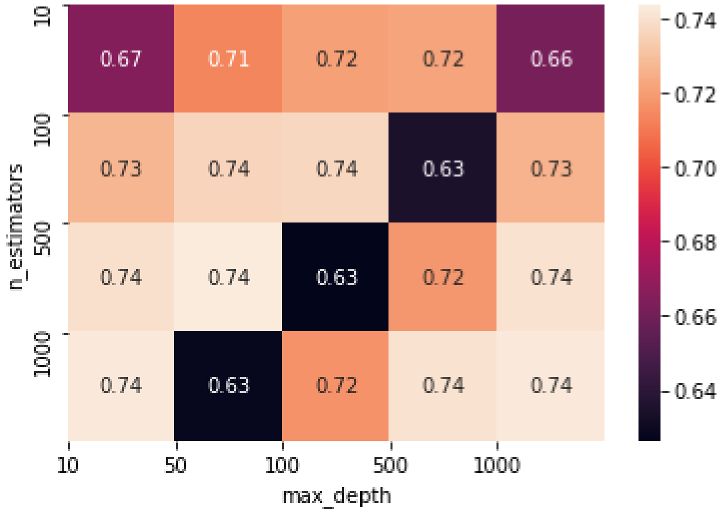
```
array([[0.66554648, 0.7138115 , 0.72087407, 0.72237968, 0.66174668],
       [0.72731306, 0.73640859, 0.73747429, 0.63467951, 0.72649105],
       [0.73986937, 0.74341672, 0.62640168, 0.71835251, 0.7404963 ],
       [0.74257345, 0.62855734, 0.71707977, 0.74040183, 0.74291331]])
```

In [6]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [7]:

```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(cv_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```

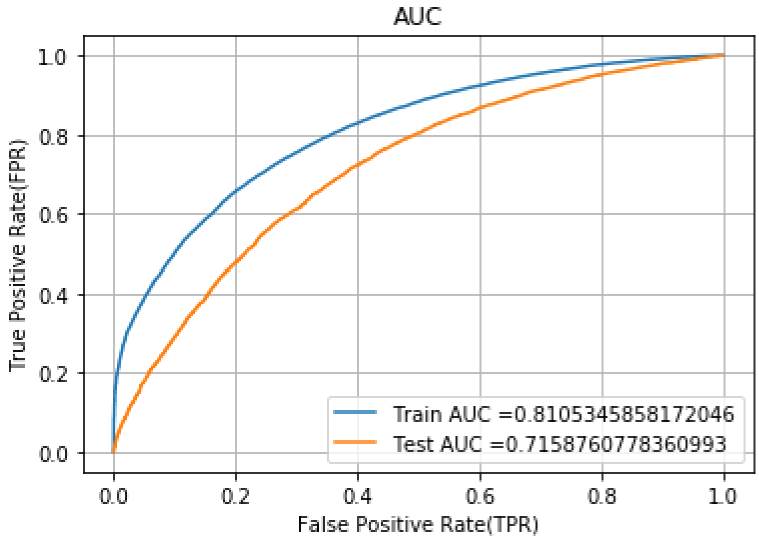


In [8]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [9]:

```
from sklearn.metrics import roc_curve, auc

model = RandomForestClassifier(max_depth = 10, n_estimators = 500,n_jobs=-1,cl
ass_weight='balanced')
model.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estim
ates of the positiveclass
# not the predicted outputs

y_train_pred = batch_predict(model, X_tr)
y_test_pred = batch_predict(model, 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_tp
r)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [10]:

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

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

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

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

In [11]:

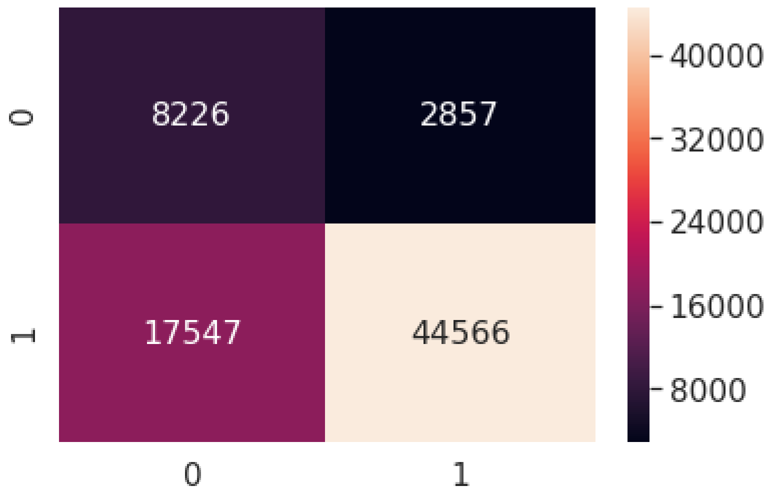
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
==============================================================================
=======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5325403533525752 for threshold 0.501
[[ 8226  2857]
 [17547 44566]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4368009743197459 for threshold 0.511
[[ 3893  1566]
 [12298 18295]]
```

In [12]:
```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_trai
n_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.5325403533525752 for threshold 0.501
```

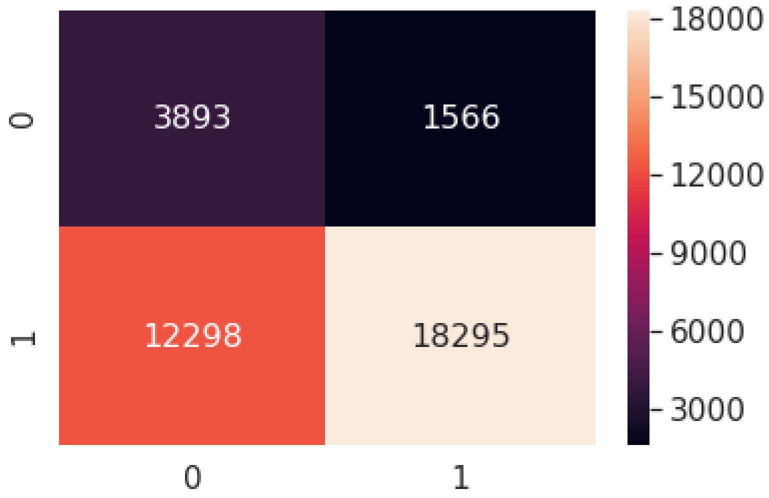
Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb64d158128>
```



In [13]:
```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_p
red, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4368009743197459 for threshold 0.511
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb63729d358>
```



In [ ]:
```

```

In [0]:
```
# Please write all the code with proper documentation
```

2.4.2 Applying Random Forests on TFIDF, SET 2

In [1]:

```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

In [3]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((cat_0_train, cat_1_train, subcat_0_train, subcat_1_train, state
_0_train, state_1_train
               ,teacher_prefix_0_train,teacher_prefix_1_train, proj_grade_0_tr
ain, proj_grade_1_train
               ,price_standardized_train,quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,text_tfidf_train,title_tfidf_train)).tocsr()
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_
one_hot_cv,teacher_prefix_one_hot_cv
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quan
tity_standardized_cv
#                ,teacher_number_of_previously_posted_projects_standardized_c
v,text_bow_cv,title_bow_cv)).tocsr()
X_te = hstack((cat_0_test, cat_1_test, subcat_0_test, subcat_1_test, state_0_t
est, state_1_test
               ,teacher_prefix_0_test,teacher_prefix_1_test,proj_grade_0_test,
proj_grade_1_test
               ,price_standardized_test,quantity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,text_tfidf_test,title_tfidf_test)).tocsr()

print("Final Data matrix on BOW")
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 on BOW
(73196, 7652) (73196,)
(36052, 7652) (36052,)
=============================================================================
=======================
```

In [4]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import time


start_time = time.time()
rf = RandomForestClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[10, 50, 100,
500, 1000]}
clf = GridSearchCV(rf, parameters, cv= 3, scoring='roc_auc',return_train_score
=True)
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']
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 3243.538765668869 ms
```

In [5]:

```
import dill
# dill.dump_session('notebook_env2.db')
dill.load_session('notebook_env2.db')
```

In [6]:

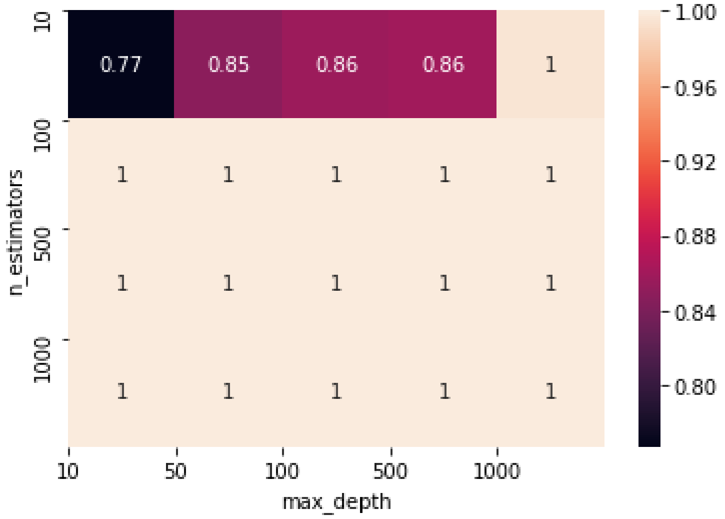
```
train_auc = train_auc.reshape(4,5)
cv_auc = cv_auc.reshape(4,5)
train_auc
cv_auc
```

Out[6]:

```
array([[0.66795936, 0.7119164 , 0.72057929, 0.72338163, 0.64496068],
       [0.71721586, 0.73282717, 0.7349762 , 0.61250366, 0.71007392],
       [0.73467203, 0.73734021, 0.61925128, 0.70644612, 0.73305086],
       [0.73657911, 0.61303282, 0.70711712, 0.73349205, 0.73621055]])
```

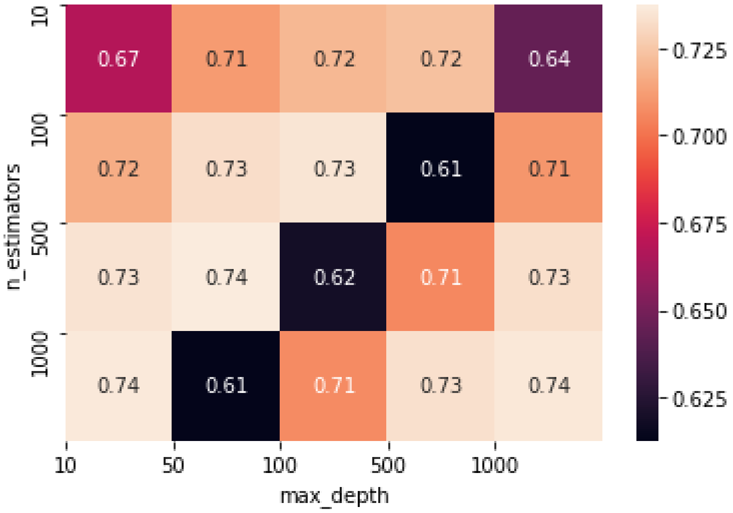
In [7]:

```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [8]:

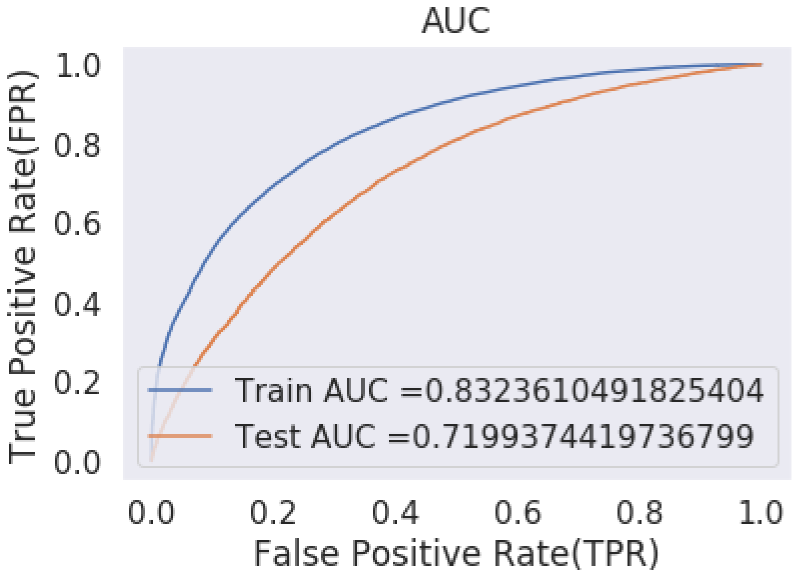
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(cv_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [10]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [55]:
```
from sklearn.metrics import roc_curve, auc

model = RandomForestClassifier(max_depth = 10, n_estimators = 1000,n_jobs=-1,c
lass_weight='balanced')
model.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estim
ates of the positiveclass
# not the predicted outputs

y_train_pred = batch_predict(model, X_tr)
y_test_pred = batch_predict(model, 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_tp
r)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [56]:

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

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

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

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

In [57]:

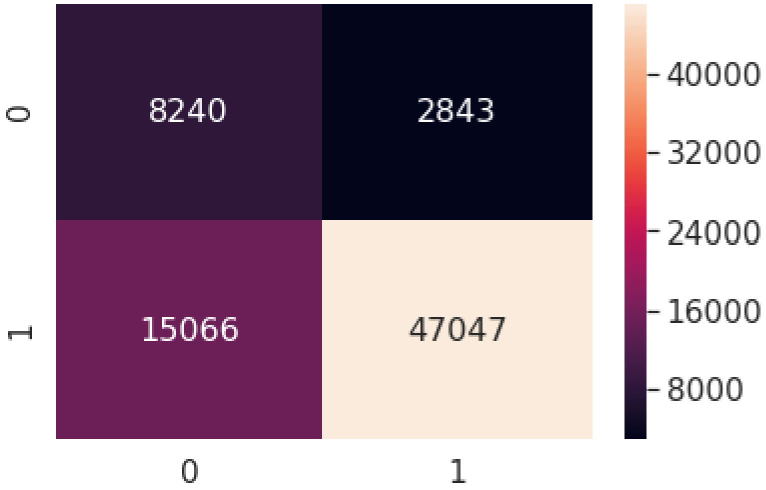
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
============================================================================
=======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5631438013598228 for threshold 0.502
[[ 8240  2843]
 [15066 47047]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.44306974645348646 for threshold 0.512
[[ 3763  1696]
 [11246 19347]]
```

In [58]:
```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.5631438013598228 for threshold 0.502
```

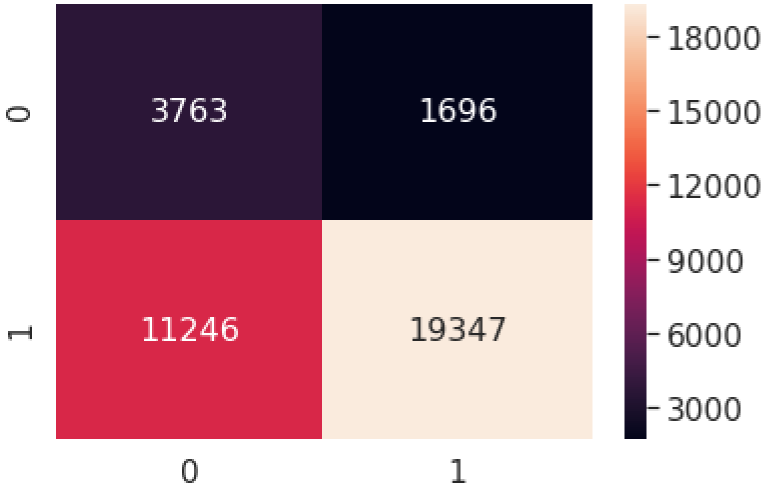
Out[58]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9394793cf8>
```



In [59]:
```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.44306974645348646 for threshold 0.512
```

Out[59]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93947b28d0>
```



In [0]:
```
# Please write all the code with proper documentation
```

2.4.3 Applying Random Forests on AVG W2V, SET 3

In [2]:
```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

In [3]:
```
avg_w2v_essays_vectors_train = np.array(avg_w2v_essays_vectors_train)
avg_w2v_titles_vectors_train = np.array(avg_w2v_titles_vectors_train)
```

In [4]:
```
avg_w2v_essays_vectors_test = np.array(avg_w2v_essays_vectors_test)
avg_w2v_titles_vectors_test = np.array(avg_w2v_titles_vectors_test)
```

In [ ]:

In [5]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = np.hstack((cat_0_train, cat_1_train, subcat_0_train, subcat_1_train, st
ate_0_train, state_1_train
               ,teacher_prefix_0_train,teacher_prefix_1_train, proj_grade_0_tr
ain, proj_grade_1_train
               ,price_standardized_train,quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,avg_w2v_essays_vectors_train
               ,avg_w2v_titles_vectors_train))
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_
one_hot_cv,teacher_prefix_one_hot_cv
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quan
tity_standardized_cv
#                ,teacher_number_of_previously_posted_projects_standardized_c
v,text_bow_cv,title_bow_cv)).tocsr()
X_te = np.hstack((cat_0_test, cat_1_test, subcat_0_test, subcat_1_test, state_
0_test, state_1_test
               ,teacher_prefix_0_test,teacher_prefix_1_test,proj_grade_0_test,
proj_grade_1_test
               ,price_standardized_test,quantity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,avg_w2v_essays_vectors_test
               ,avg_w2v_titles_vectors_test))

print("Final Data matrix on BOW")
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 on BOW
(73196, 613) (73196,)
(36052, 613) (36052,)
=============================================================================
=======================
```

In [6]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import time


start_time = time.time()
rf = RandomForestClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[10, 50, 100,
500, 1000]}
clf = GridSearchCV(rf, parameters, cv= 3, scoring='roc_auc',return_train_score
=True)
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']
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 3207.920234441757 ms
```

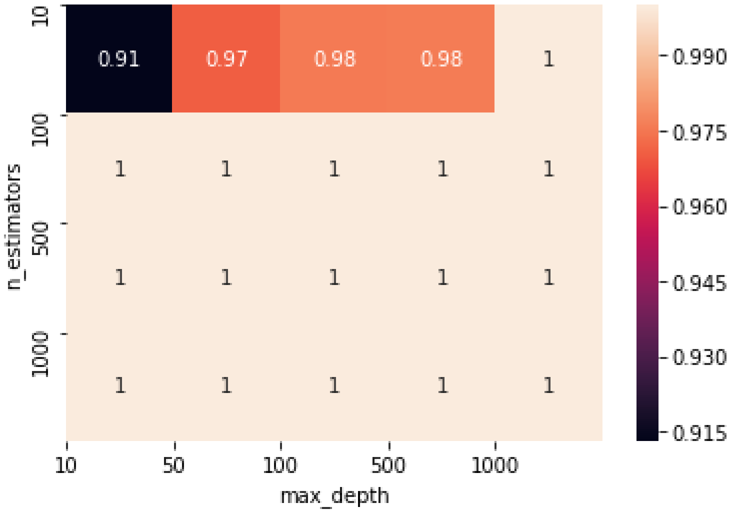
In [2]:
```
import dill
# dill.dump_session('notebook_env3.db')
dill.load_session('notebook_env3.db')
```

In [3]:
```
train_auc = train_auc.reshape(4,5)
cv_auc = cv_auc.reshape(4,5)
train_auc
cv_auc
```

Out[3]:
```
array([[0.65555785, 0.70523334, 0.7129245 , 0.71478654, 0.58488636],
       [0.67039123, 0.69789432, 0.70089872, 0.58483381, 0.67034646],
       [0.69720824, 0.70058421, 0.58746747, 0.67194759, 0.69970504],
       [0.70271764, 0.58574051, 0.67228709, 0.69688772, 0.70236634]])
```

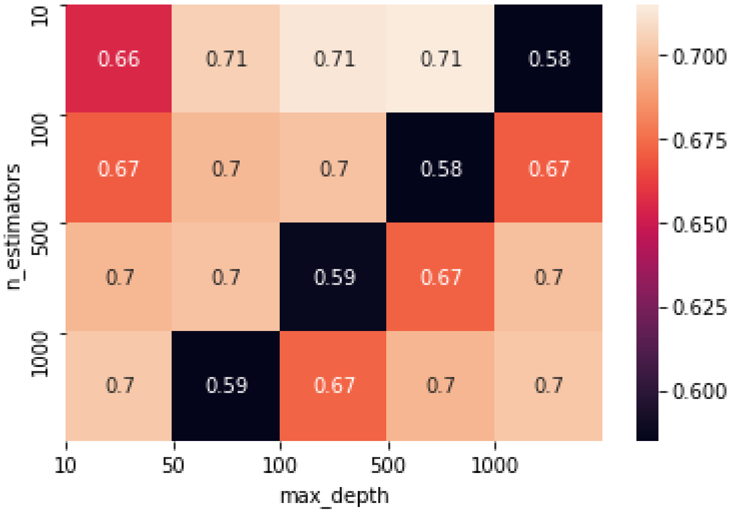
In [4]:

```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [5]:

```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(cv_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```

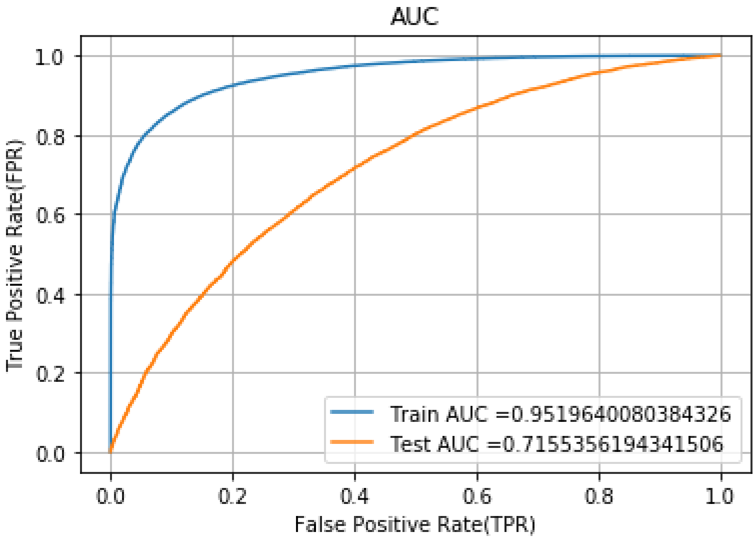


In [6]:

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

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

In [7]:

```
from sklearn.metrics import roc_curve, auc

model = RandomForestClassifier(max_depth = 10, n_estimators = 500,n_jobs=-1,class_weight='balanced')
model.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positiveclass
# not the predicted outputs

y_train_pred = batch_predict(model, X_tr)
y_test_pred = batch_predict(model, 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("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [8]:

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

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

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

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

In [9]:

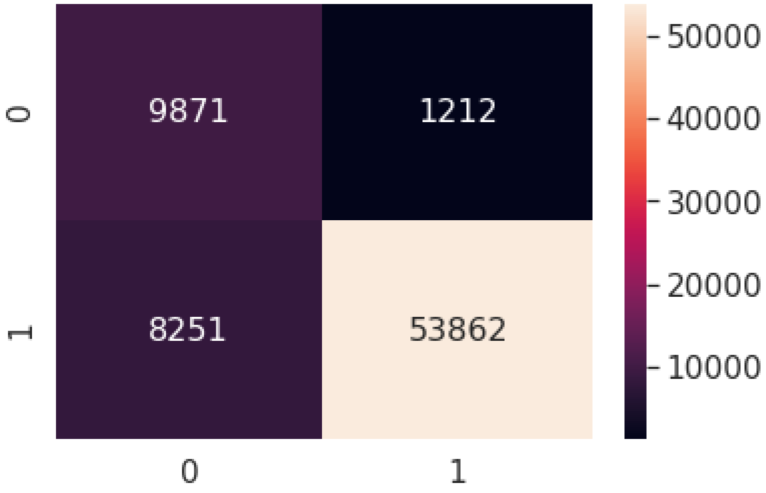
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
==============================================================================
=======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.7723315716872133 for threshold 0.532
[[ 9871  1212]
 [ 8251 53862]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4325839941247558 for threshold 0.506
[[ 2024  3435]
 [ 3606 26987]]
```

In [10]:
```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.7723315716872133 for threshold 0.532
```

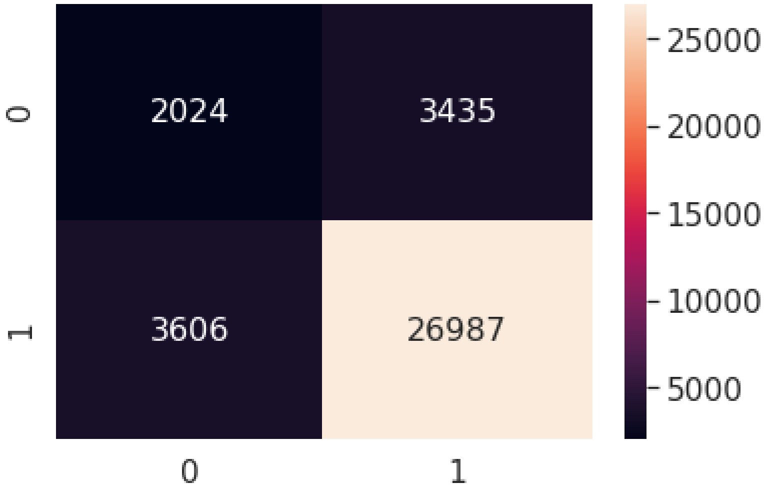
Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a904c9eb8>
```



In [11]:
```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4325839941247558 for threshold 0.506
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a90509dd8>
```



In [0]:
```
# Please write all the code with proper documentation
```

2.4.4 Applying Random Forests on TFIDF W2V, SET 4

In [1]:
```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

In [2]:
```
tfidf_w2v_essays_vectors_train = np.array(tfidf_w2v_essays_vectors_train)
tfidf_w2v_titles_vectors_train = np.array(tfidf_w2v_titles_vectors_train)
```

In [3]:
```
tfidf_w2v_essays_vectors_test = np.array(tfidf_w2v_essays_vectors_test)
tfidf_w2v_titles_vectors_test = np.array(tfidf_w2v_titles_vectors_test)
```

In [ ]:

In [4]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = np.hstack((cat_0_train, cat_1_train, subcat_0_train, subcat_1_train, st
ate_0_train, state_1_train
               ,teacher_prefix_0_train,teacher_prefix_1_train, proj_grade_0_tr
ain, proj_grade_1_train
               ,price_standardized_train,quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,tfidf_w2v_essays_vectors_train
               ,tfidf_w2v_titles_vectors_train))
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_
one_hot_cv,teacher_prefix_one_hot_cv
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quan
tity_standardized_cv
#                ,teacher_number_of_previously_posted_projects_standardized_c
v,text_bow_cv,title_bow_cv)).tocsr()
X_te = np.hstack((cat_0_test, cat_1_test, subcat_0_test, subcat_1_test, state_
0_test, state_1_test
               ,teacher_prefix_0_test,teacher_prefix_1_test,proj_grade_0_test,
proj_grade_1_test
               ,price_standardized_test,quantity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,tfidf_w2v_essays_vectors_test
               ,tfidf_w2v_titles_vectors_test))

print("Final Data matrix on BOW")
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 on BOW
(73196, 613) (73196,)
(36052, 613) (36052,)
=============================================================================
=======================
```

In [5]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
import time


start_time = time.time()
rf = RandomForestClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[10, 50, 100,
500, 1000]}
clf = GridSearchCV(rf, parameters, cv= 3, scoring='roc_auc',return_train_score
=True)
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']
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 3181.807500600815 ms
```

In [6]:

```
import dill
# dill.dump_session('notebook_env4.db')
# dill.load_session('notebook_env4.db')
```

In [7]:

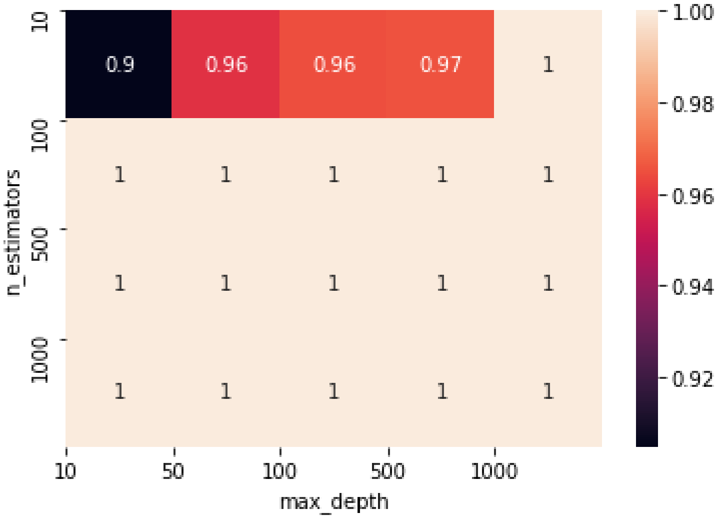
```
train_auc = train_auc.reshape(4,5)
cv_auc = cv_auc.reshape(4,5)
train_auc
cv_auc
```

Out[7]:

```
array([[0.65910296, 0.69830771, 0.70600712, 0.70611675, 0.5822068 ],
       [0.66849952, 0.69326687, 0.69762812, 0.58830799, 0.66771538],
       [0.69459207, 0.69755436, 0.58620884, 0.66584815, 0.69184751],
       [0.69695478, 0.59068504, 0.66927736, 0.69230991, 0.69714666]])
```

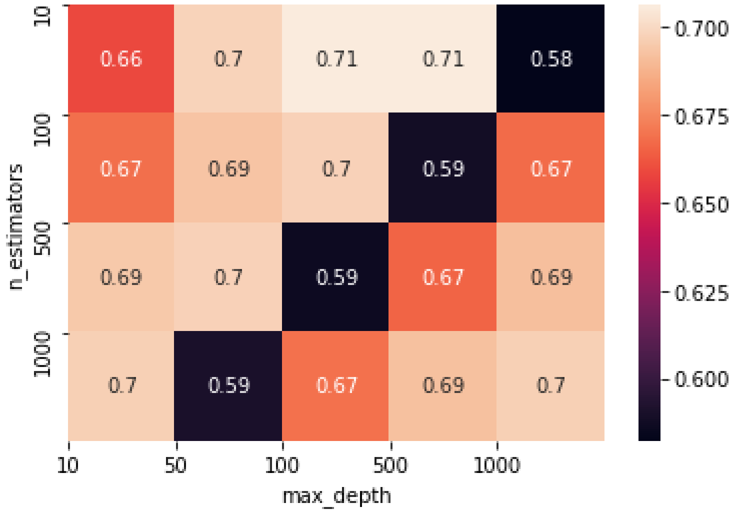
In [8]:

```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [9]:

```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(cv_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```

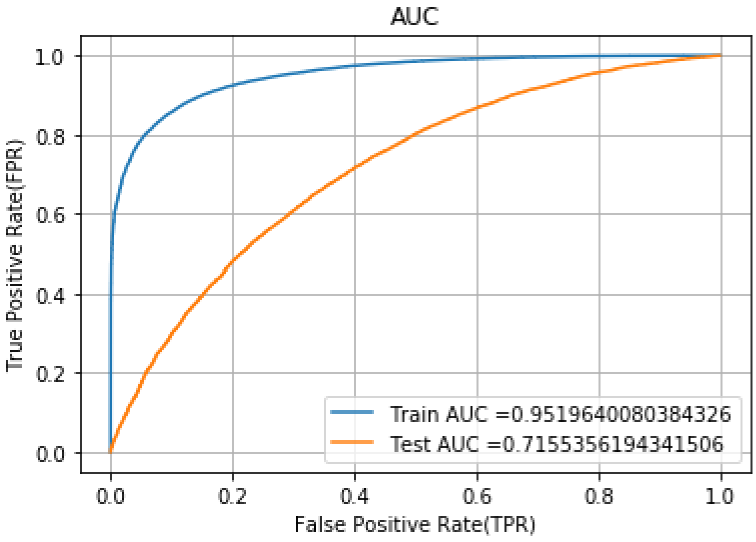


In [10]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [7]:

```python
from sklearn.metrics import roc_curve, auc

model = RandomForestClassifier(max_depth = 10, n_estimators = 500,n_jobs=-1,class_weight='balanced')
model.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positiveclass
# not the predicted outputs

y_train_pred = batch_predict(model, X_tr)
y_test_pred = batch_predict(model, 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("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [8]:

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

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

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

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

In [9]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
==============================================================================
======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.7723315716872133 for threshold 0.532
[[ 9871  1212]
 [ 8251 53862]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4325839941247558 for threshold 0.506
[[ 2024  3435]
 [ 3606 26987]]
```

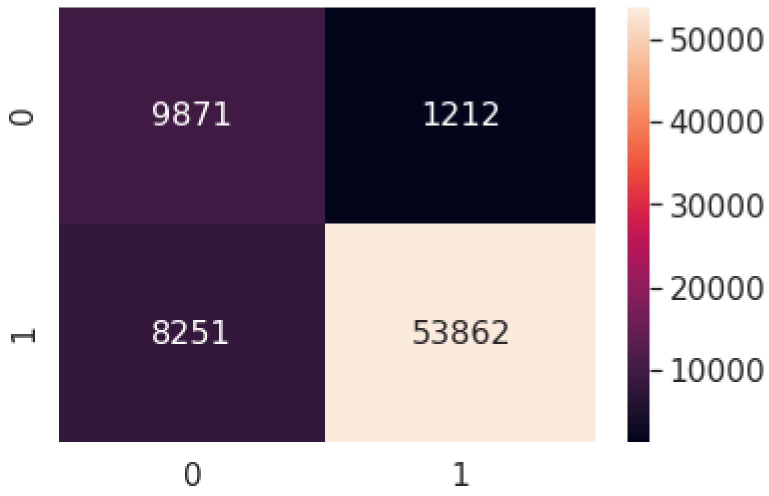
In [10]:

```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.7723315716872133 for threshold 0.532
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a904c9eb8>
```



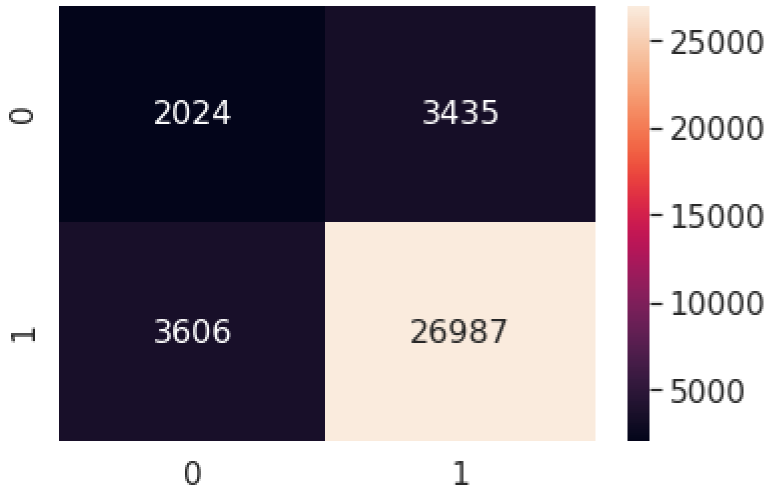
In [11]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4325839941247558 for threshold 0.506
```

Out[11]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3a90509dd8>
```



In [ ]:

In [0]:

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

2.5 Applying GBDT

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

2.5.1 Applying XGBOOST on BOW, SET 1

In [1]:

```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

In [3]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((cat_0_train, cat_1_train, subcat_0_train, subcat_1_train, state
_0_train, state_1_train
               ,teacher_prefix_0_train,teacher_prefix_1_train, proj_grade_0_tr
ain, proj_grade_1_train
               ,price_standardized_train,quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,text_bow_train,title_bow_train)).tocsr()
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_
one_hot_cv,teacher_prefix_one_hot_cv
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quan
tity_standardized_cv
#                ,teacher_number_of_previously_posted_projects_standardized_c
v,text_bow_cv,title_bow_cv)).tocsr()
X_te = hstack((cat_0_test, cat_1_test, subcat_0_test, subcat_1_test, state_0_t
est, state_1_test
               ,teacher_prefix_0_test,teacher_prefix_1_test,proj_grade_0_test,
proj_grade_1_test
               ,price_standardized_test,quantity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,text_bow_test,title_bow_test )).tocsr()

print("Final Data matrix on BOW")
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 on BOW
(73196, 7652) (73196,)
(36052, 7652) (36052,)
=============================================================================
=======================
```

In [5]:

```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
import time


start_time = time.time()
gbdt = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[10, 50, 100, 500, 1000]}
clf = GridSearchCV(gbdt, parameters, cv= 3, scoring='roc_auc',return_train_score=True)
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']
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 9235.373923301697 ms
```

In [6]:

```
import dill
# dill.dump_session('notebook_env11.db')
dill.load_session('notebook_env11.db')
```

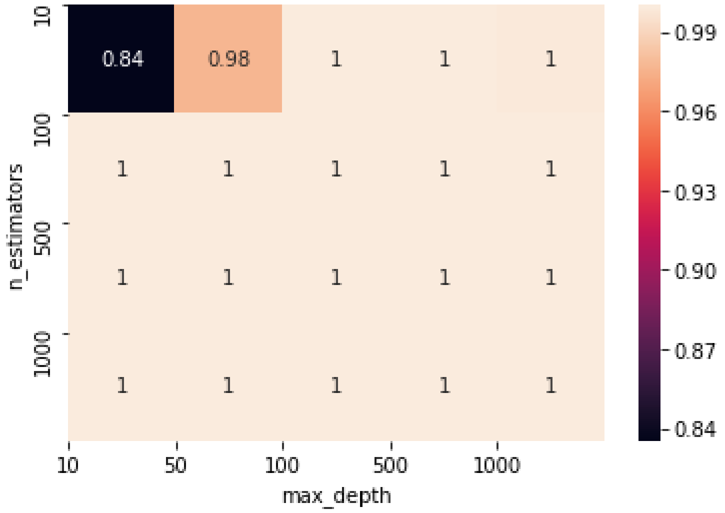
In [7]:

```
train_auc = train_auc.reshape(4,5)
cv_auc = cv_auc.reshape(4,5)
train_auc
cv_auc
```

Out[7]:

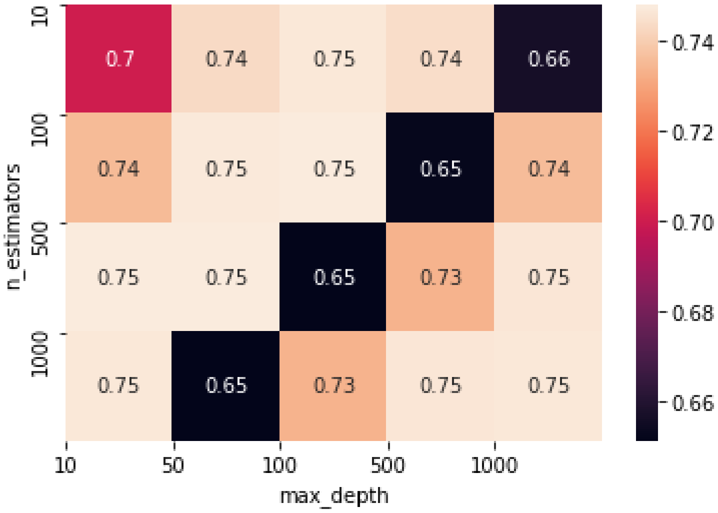
```
array([[0.70022963, 0.74347692, 0.74681136, 0.74401005, 0.656688  ],
       [0.7354928 , 0.74708909, 0.7474287 , 0.65236077, 0.73588217],
       [0.74770391, 0.74759603, 0.65105023, 0.73348236, 0.74621154],
       [0.74666423, 0.65105023, 0.73348236, 0.74621154, 0.74666423]])
```

In [8]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [9]:

```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(cv_auc,annot=True)

plt.yticks(np.arange(4), [10, 100, 500, 1000])
plt.xticks(np.arange(5), [10, 50, 100, 500, 1000])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```

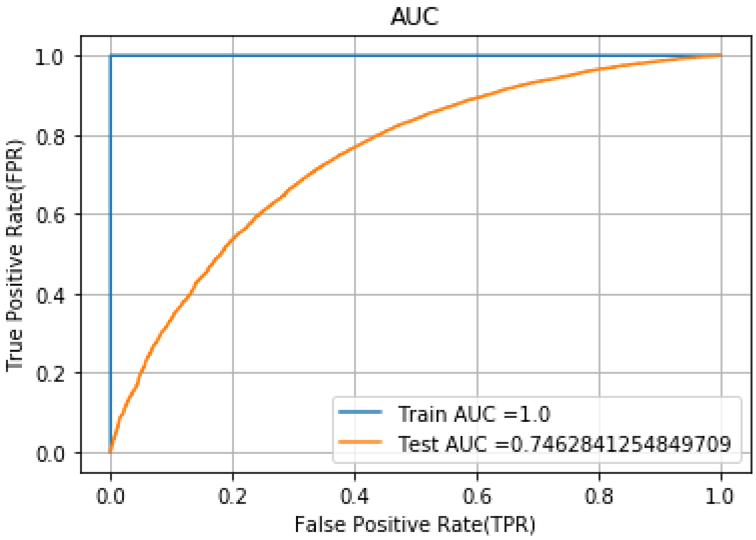


In [10]:

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [11]:

```
from sklearn.metrics import roc_curve, auc

gbdt = xgb.XGBClassifier(max_depth = 50, n_estimators = 500,n_jobs=-1,class_we
ight='balanced')
gbdt.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estim
ates of the positiveclass
# not the predicted outputs

y_train_pred = batch_predict(gbdt, X_tr)
y_test_pred = batch_predict(gbdt, 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_tp
r)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [12]:

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

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

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

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

In [13]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
=============================================================================
=======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 1.0 for threshold 0.738
[[11083     0]
 [    0 62113]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4717939474065868 for threshold 0.999
[[ 5104   355]
 [22980  7613]]
```

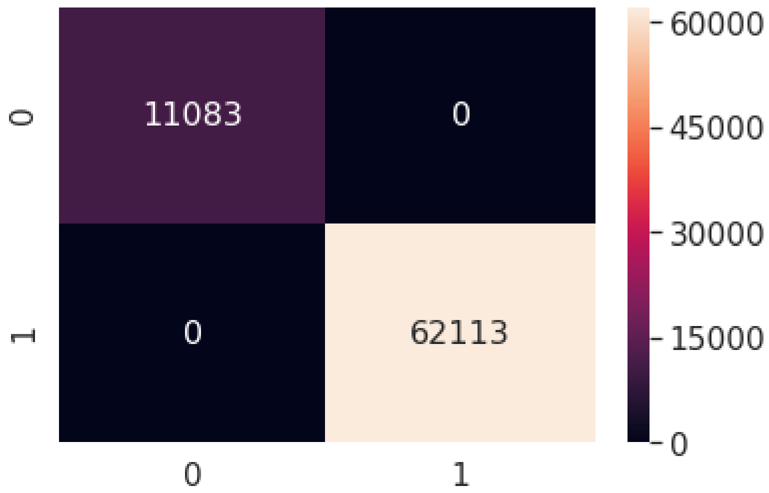
In [14]:

```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 1.0 for threshold 0.738
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06c8e3a2b0>
```



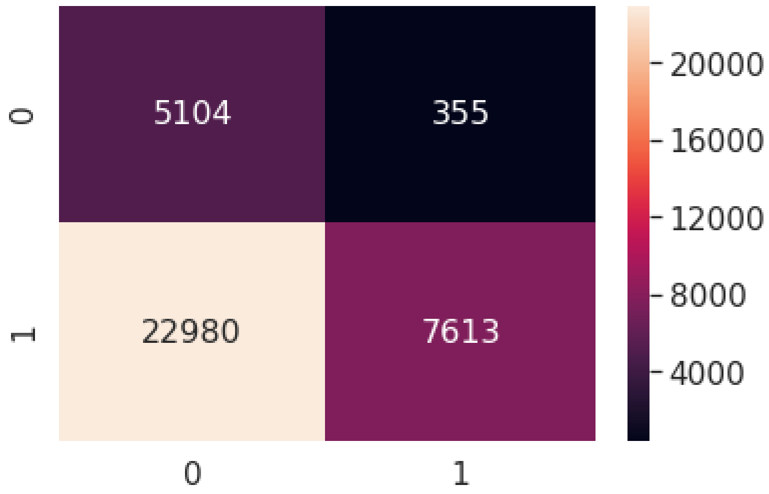
In [15]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4717939474065868 for threshold 0.999
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06c8c56860>
```



In [ ]:

In [0]:

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

2.5.2 Applying XGBOOST on TFIDF, SET 2

In [1]:

```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

In [2]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((cat_0_train, cat_1_train, subcat_0_train, subcat_1_train, state
_0_train, state_1_train
               ,teacher_prefix_0_train,teacher_prefix_1_train, proj_grade_0_tr
ain, proj_grade_1_train
               ,price_standardized_train,quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,text_tfidf_train,title_tfidf_train)).tocsr()
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_
one_hot_cv,teacher_prefix_one_hot_cv
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quan
tity_standardized_cv
#                ,teacher_number_of_previously_posted_projects_standardized_c
v,text_bow_cv,title_bow_cv)).tocsr()
X_te = hstack((cat_0_test, cat_1_test, subcat_0_test, subcat_1_test, state_0_t
est, state_1_test
               ,teacher_prefix_0_test,teacher_prefix_1_test,proj_grade_0_test,
proj_grade_1_test
               ,price_standardized_test,quantity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,text_tfidf_test,title_tfidf_test)).tocsr()

print("Final Data matrix on BOW")
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 on BOW
(73196, 7652) (73196,)
(36052, 7652) (36052,)
==============================================================================
======================
```

In [3]:

```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
import time

start_time = time.time()
gbdt = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 500], 'max_depth':[10, 50, 100, 500]}
clf = GridSearchCV(gbdt, parameters, cv= 3, scoring='roc_auc',return_train_sco
re=True)
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']
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 5943.0765879154205 ms
```

In [1]:

```
import dill
# dill.dump_session('notebook_env22.db')
# dill.load_session('notebook_env22.db')
```

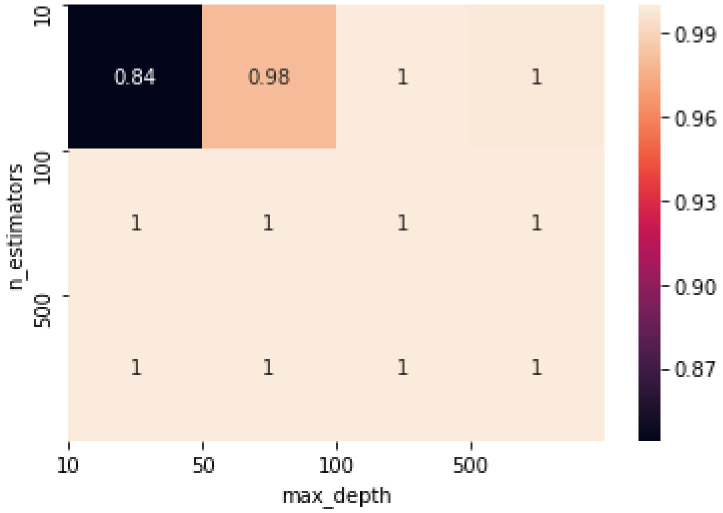
In [2]:

```
train_auc = train_auc.reshape(3,4)
cv_auc = cv_auc.reshape(3,4)
train_auc
cv_auc
```

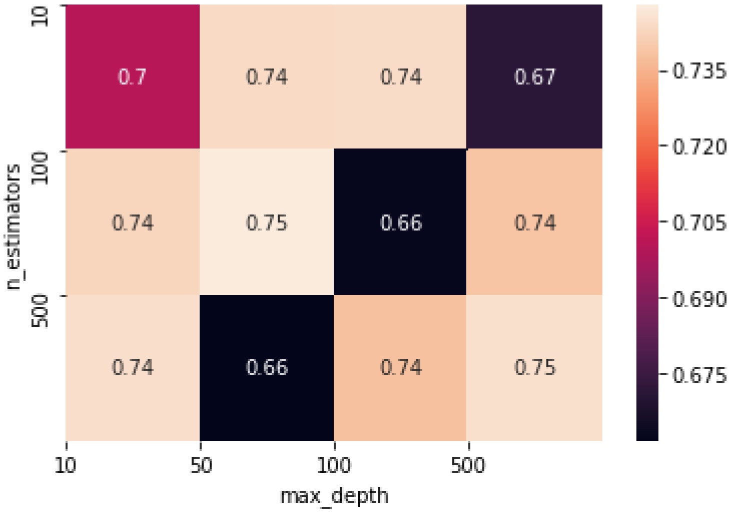
Out[2]:

```
array([[0.70063637, 0.74394347, 0.74411849, 0.67142887],
       [0.74277923, 0.74770023, 0.66278156, 0.73865109],
       [0.74483859, 0.66169776, 0.73816269, 0.74532923]])
```

In [3]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(3), [10, 100, 500])
plt.xticks(np.arange(4), [10, 50, 100, 500])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```

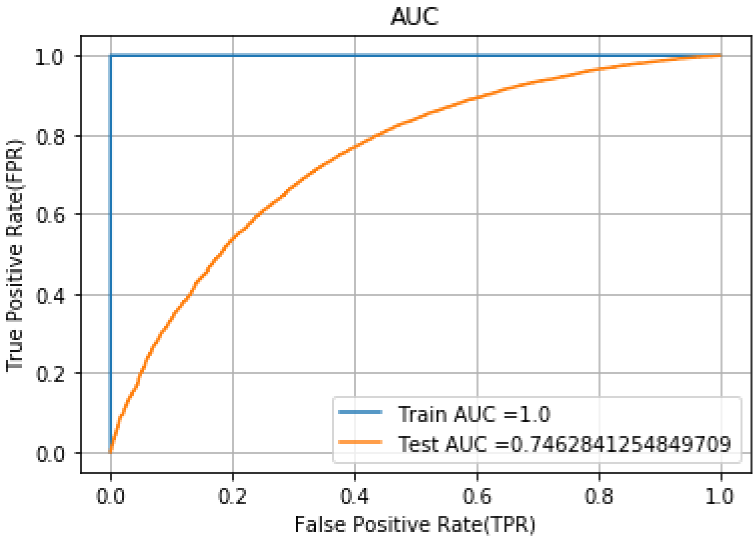


In [4]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(cv_auc,annot=True)

plt.yticks(np.arange(3), [10, 100, 500])
plt.xticks(np.arange(4), [10, 50, 100, 500])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [10]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [11]:
```
from sklearn.metrics import roc_curve, auc

gbdt = xgb.XGBClassifier(max_depth = 50, n_estimators = 500,n_jobs=-1,class_we
ight='balanced')
gbdt.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estim
ates of the positiveclass
# not the predicted outputs

y_train_pred = batch_predict(gbdt, X_tr)
y_test_pred = batch_predict(gbdt, 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_tp
r)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [12]:

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

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

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

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

In [13]:

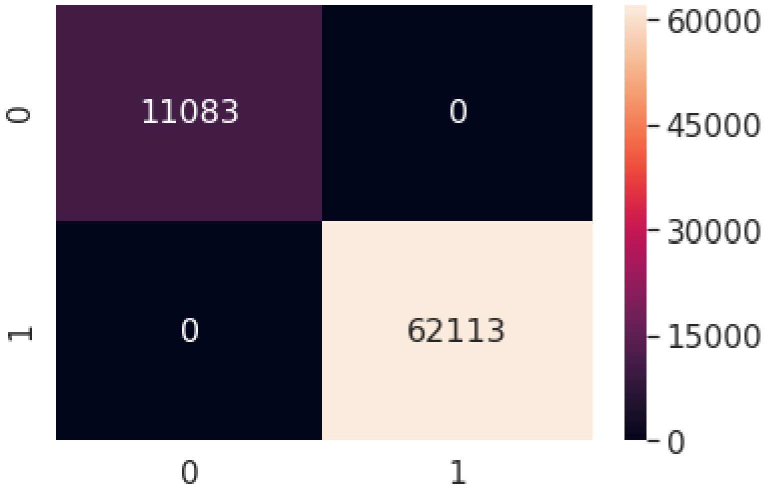
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
=============================================================================
=======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 1.0 for threshold 0.738
[[11083     0]
 [    0 62113]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4717939474065868 for threshold 0.999
[[ 5104   355]
 [22980  7613]]
```

In [14]:
```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 1.0 for threshold 0.738
```

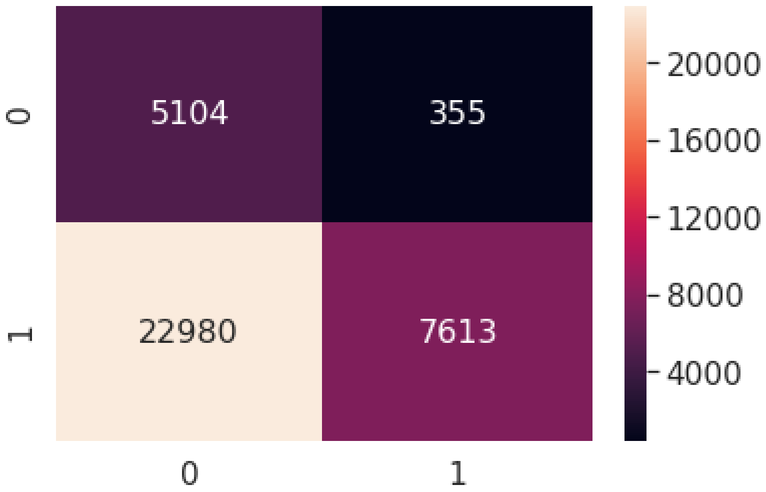
Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06c8e3a2b0>
```



In [15]:
```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4717939474065868 for threshold 0.999
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f06c8c56860>
```



In [ ]:

In [0]:
```
# Please write all the code with proper documentation
```

2.5.3 Applying XGBOOST on AVG W2V, SET 3

In [5]:

```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

In [6]:

```
avg_w2v_essays_vectors_train = np.array(avg_w2v_essays_vectors_train)
avg_w2v_titles_vectors_train = np.array(avg_w2v_titles_vectors_train)
```

In [7]:

```
avg_w2v_essays_vectors_test = np.array(avg_w2v_essays_vectors_test)
avg_w2v_titles_vectors_test = np.array(avg_w2v_titles_vectors_test)
```

In [ ]:

In [8]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = np.hstack((cat_0_train, cat_1_train, subcat_0_train, subcat_1_train, st
ate_0_train, state_1_train
               ,teacher_prefix_0_train,teacher_prefix_1_train, proj_grade_0_tr
ain, proj_grade_1_train
               ,price_standardized_train,quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,avg_w2v_essays_vectors_train
               ,avg_w2v_titles_vectors_train))
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_
one_hot_cv,teacher_prefix_one_hot_cv
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quan
tity_standardized_cv
#                ,teacher_number_of_previously_posted_projects_standardized_c
v,text_bow_cv,title_bow_cv)).tocsr()
X_te = np.hstack((cat_0_test, cat_1_test, subcat_0_test, subcat_1_test, state_
0_test, state_1_test
               ,teacher_prefix_0_test,teacher_prefix_1_test,proj_grade_0_test,
proj_grade_1_test
               ,price_standardized_test,quantity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,avg_w2v_essays_vectors_test
               ,avg_w2v_titles_vectors_test))

print("Final Data matrix on BOW")
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 on BOW
(73196, 613) (73196,)
(36052, 613) (36052,)
============================================================================
========================
```

In [9]:

```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
import time

start_time = time.time()
gbdt = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 500], 'max_depth':[10, 50, 100, 500]}
clf = GridSearchCV(gbdt, parameters, cv= 3, scoring='roc_auc',return_train_sco
re=True)
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']
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 6477.814277648926 ms
```

In [1]:

```
import dill
# dill.dump_session('notebook_env23.db')
dill.load_session('notebook_env23.db')
```

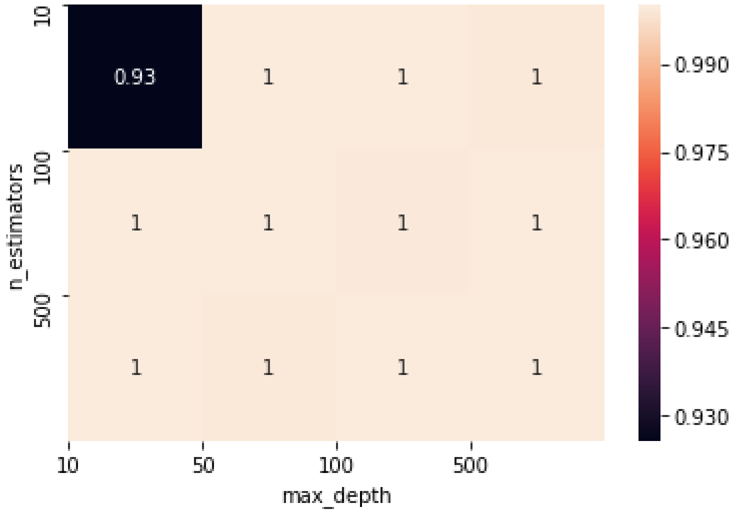
In [2]:

```
train_auc = train_auc.reshape(3,4)
cv_auc = cv_auc.reshape(3,4)
train_auc
cv_auc
```

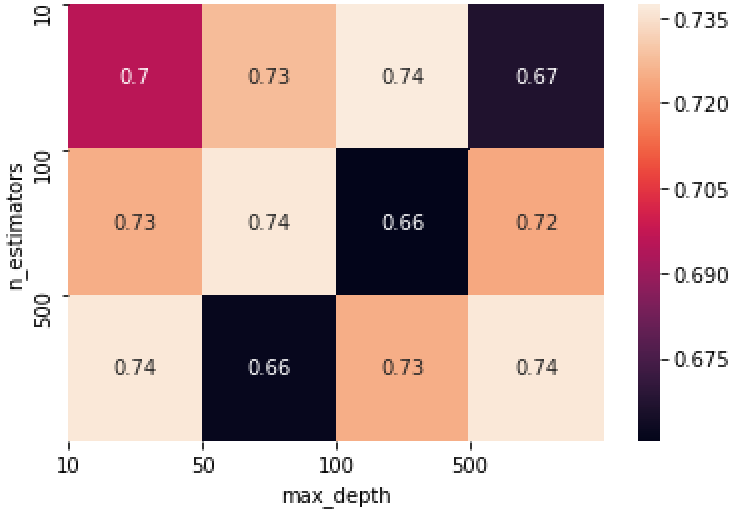
Out[2]:

```
array([[0.69561052, 0.72769529, 0.73731143, 0.66685636],
       [0.72520949, 0.73658591, 0.6603649 , 0.72395276],
       [0.73649141, 0.66177614, 0.72513686, 0.73652392]])
```

In [3]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(3), [10, 100, 500])
plt.xticks(np.arange(4), [10, 50, 100, 500])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```

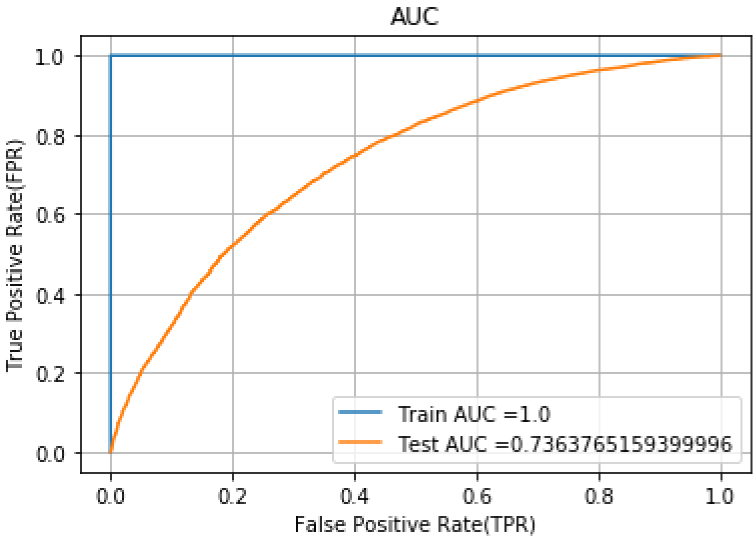


In [4]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(cv_auc,annot=True)

plt.yticks(np.arange(3), [10, 100, 500])
plt.xticks(np.arange(4), [10, 50, 100, 500])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [5]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

```
In [6]: from sklearn.metrics import roc_curve, auc

        gbdt = xgb.XGBClassifier(max_depth = 100, n_estimators = 500,n_jobs=-1,class_w
        eight='balanced')
        gbdt.fit(X_tr, y_train)
        # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estim
        ates of the positiveclass
        # not the predicted outputs

        y_train_pred = batch_predict(gbdt, X_tr)
        y_test_pred = batch_predict(gbdt, 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_tp
        r)))
        plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
        plt.legend()
        plt.xlabel("False Positive Rate(TPR)")
        plt.ylabel("True Positive Rate(FPR)")
        plt.title("AUC")
        plt.grid()
        plt.show()
```



In [7]:

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

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

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

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

In [8]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
============================================================================
=======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 1.0 for threshold 0.847
[[11083     0]
 [    0 62113]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.45608344986973526 for threshold 0.999
[[ 4935   524]
 [21125  9468]]
```

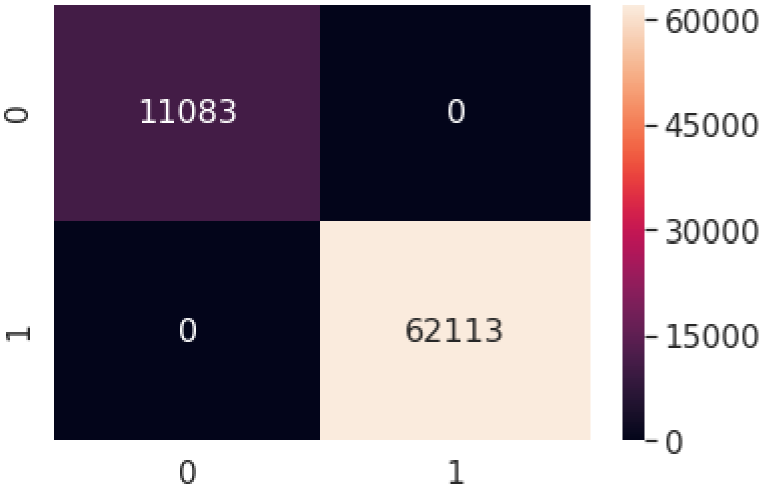
In [9]:

```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 1.0 for threshold 0.847
```

Out[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ffc868a58>
```



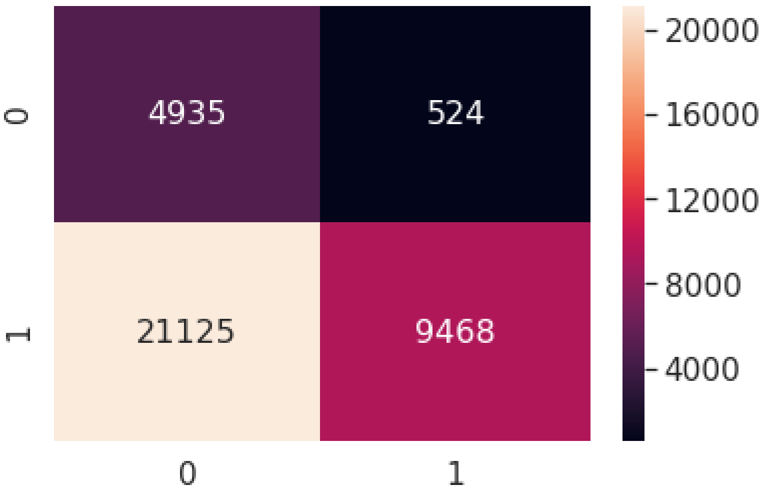
In [10]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.45608344986973526 for threshold 0.999
```

Out[10]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ffc7e2390>
```



In [ ]:

In [0]:

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

2.5.4 Applying XGBOOST on TFIDF W2V, SET 4

In [1]:
```
import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

In [2]:
```
tfidf_w2v_essays_vectors_train = np.array(tfidf_w2v_essays_vectors_train)
tfidf_w2v_titles_vectors_train = np.array(tfidf_w2v_titles_vectors_train)
```

In [3]:
```
tfidf_w2v_essays_vectors_test = np.array(tfidf_w2v_essays_vectors_test)
tfidf_w2v_titles_vectors_test = np.array(tfidf_w2v_titles_vectors_test)
```

In [ ]:
```

```

In [4]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = np.hstack((cat_0_train, cat_1_train, subcat_0_train, subcat_1_train, st
ate_0_train, state_1_train
               ,teacher_prefix_0_train,teacher_prefix_1_train, proj_grade_0_tr
ain, proj_grade_1_train
               ,price_standardized_train,quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_trai
n,tfidf_w2v_essays_vectors_train
               ,tfidf_w2v_titles_vectors_train))
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_
one_hot_cv,teacher_prefix_one_hot_cv
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quan
tity_standardized_cv
#                ,teacher_number_of_previously_posted_projects_standardized_c
v,text_bow_cv,title_bow_cv)).tocsr()
X_te = np.hstack((cat_0_test, cat_1_test, subcat_0_test, subcat_1_test, state_
0_test, state_1_test
               ,teacher_prefix_0_test,teacher_prefix_1_test,proj_grade_0_test,
proj_grade_1_test
               ,price_standardized_test,quantity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test
,tfidf_w2v_essays_vectors_test
               ,tfidf_w2v_titles_vectors_test))

print("Final Data matrix on BOW")
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 on BOW
(73196, 613) (73196,)
(36052, 613) (36052,)
==============================================================================
======================
```

In [6]:

```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
import time

start_time = time.time()
gbdt = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 500], 'max_depth':[10, 50, 100, 500]}
clf = GridSearchCV(gbdt, parameters, cv= 3, scoring='roc_auc',return_train_sco
re=True)
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']
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 6498.283704280853 ms
```

In [7]:

```
import dill
# dill.dump_session('notebook_env24.db')
# dill.load_session('notebook_env24.db')
```

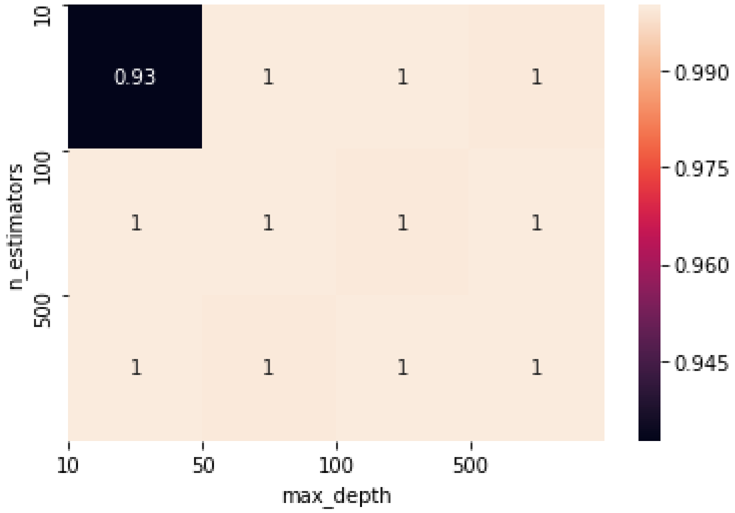
In [8]:

```
train_auc = train_auc.reshape(3,4)
cv_auc = cv_auc.reshape(3,4)
train_auc
cv_auc
```

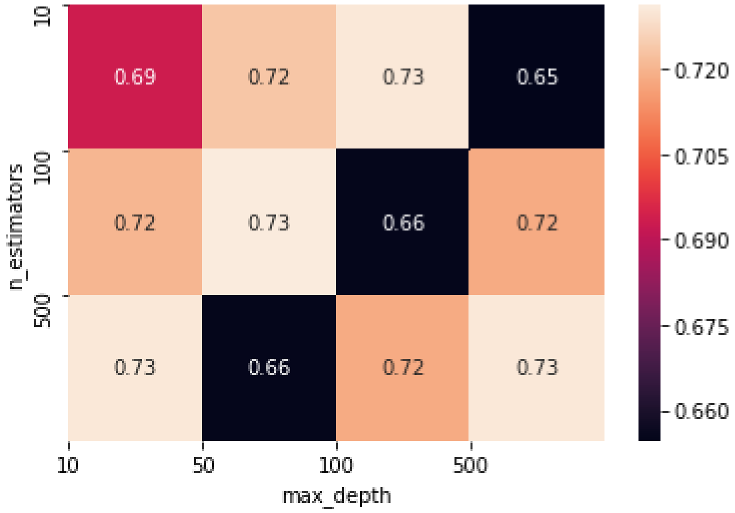
Out[8]:

```
array([[0.69320481, 0.72367601, 0.7304125 , 0.65460101],
       [0.72071682, 0.7311896 , 0.65524462, 0.71841107],
       [0.73032627, 0.65524462, 0.71841107, 0.73032627]])
```

In [9]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(3), [10, 100, 500])
plt.xticks(np.arange(4), [10, 50, 100, 500])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```

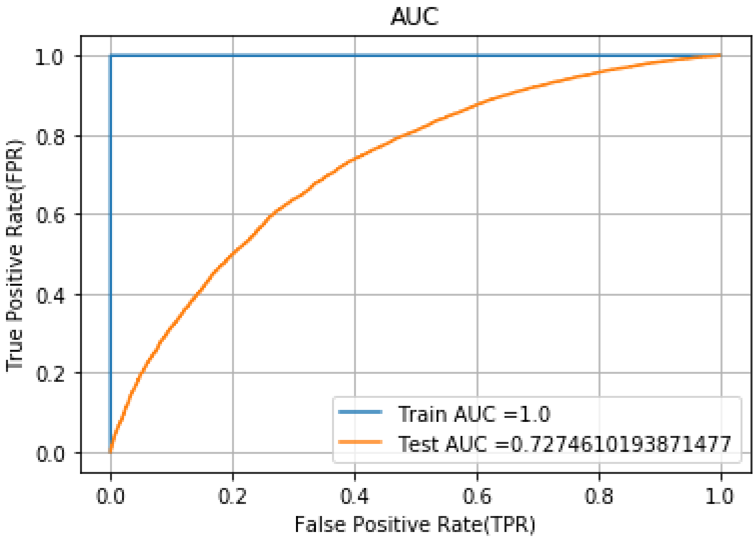


In [10]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(cv_auc,annot=True)

plt.yticks(np.arange(3), [10, 100, 500])
plt.xticks(np.arange(4), [10, 50, 100, 500])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [12]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [13]:
```
from sklearn.metrics import roc_curve, auc

gbdt = xgb.XGBClassifier(max_depth = 10, n_estimators = 500,n_jobs=-1,class_we
ight='balanced')
gbdt.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estim
ates of the positiveclass
# not the predicted outputs

y_train_pred = batch_predict(gbdt, X_tr)
y_test_pred = batch_predict(gbdt, 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_tp
r)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate(TPR)")
plt.ylabel("True Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [14]:

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

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

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

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

In [15]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
============================================================================
=======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 1.0 for threshold 0.703
[[11083     0]
 [    0 62113]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4499805747880779 for threshold 0.999
[[ 5206   253]
 [24972  5621]]
```

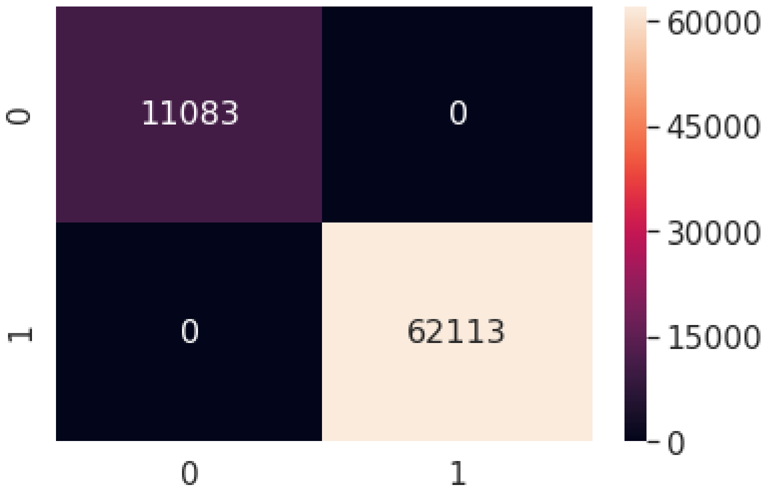
In [16]:

```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 1.0 for threshold 0.703
```

Out[16]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c34557828>
```



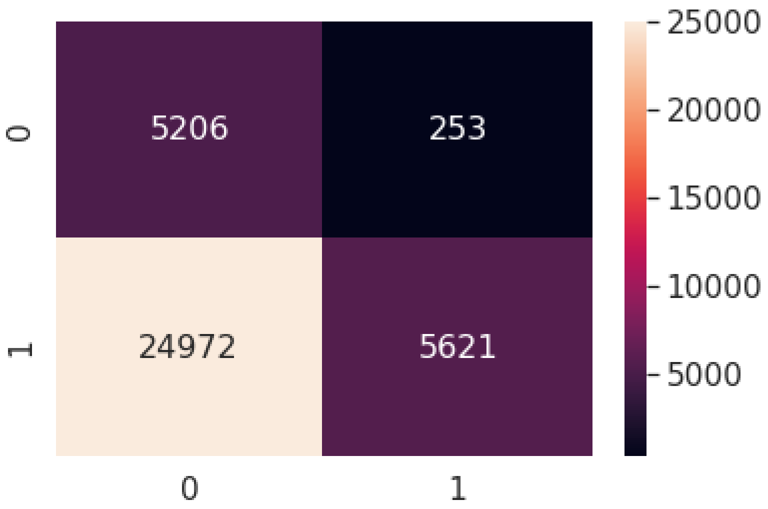
In [17]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4499805747880779 for threshold 0.999
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4c362afbe0>
```



In [0]:

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

3. Conclusion

In [0]:

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

In [19]:

```
from prettytable import PrettyTable
x = PrettyTable()


x.field_names = ["Vectorizer", "Model", "Hyperparameters(max_depth,n_estimators)" , "Test AUC"]
x.add_row(["BOW", "RF","(10,500)", 0.7158])
x.add_row(["TFIDF", "RF", "(10,1000)", 0.7199])
x.add_row(["AVG W2V", "RF", "(10,500)", 0.7155])
x.add_row(["TFIDF W2V", "RF", "(10,500)", 0.7155])
x.add_row(["BOW", "GBDT","(50,500)", 0.7462])
x.add_row(["TFIDF", "GBDT", "(50,500)", 0.7462])
x.add_row(["AVG W2V", "GBDT", "(100,500)", 0.7363])
x.add_row(["TFIDF W2V", "GBDT", "(10,500)", 0.7274])
print(x)
```

```
+------------+-------+-----------------------------------------+----------+
| Vectorizer | Model | Hyperparameters(max_depth,n_estimators) | Test AUC |
+------------+-------+-----------------------------------------+----------+
|    BOW     |   RF  |                 (10,500)                |  0.7158  |
|   TFIDF    |   RF  |                (10,1000)                |  0.7199  |
|  AVG W2V   |   RF  |                 (10,500)                |  0.7155  |
| TFIDF W2V  |   RF  |                 (10,500)                |  0.7155  |
|    BOW     |  GBDT |                 (50,500)                |  0.7462  |
|   TFIDF    |  GBDT |                 (50,500)                |  0.7462  |
|  AVG W2V   |  GBDT |                (100,500)                |  0.7363  |
| TFIDF W2V  |  GBDT |                 (10,500)                |  0.7274  |
+------------+-------+-----------------------------------------+----------+
```

In [ ]: