DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

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

Assignment 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: use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
   - Set 2: categorical(instead of one hot encoding, try response coding: use probability values), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
   - Set 3: categorical(instead of one hot encoding, try response coding: use probability values), numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
   - Set 4: categorical(instead of one hot encoding, try response coding: 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)**
   - Consider the following range for hyperparameters **n_estimators** = [10, 50, 100, 150, 200, 300, 500, 1000], **max_depth** = [2, 3, 4, 5, 6, 7, 8, 9, 10]
   - Find the best hyper parameter which will give the maximum AUC 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

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 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 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

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-leakage, 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.

1.1 Reading Data

In [2]:

```
project_data=pd.read_csv('train_data.csv', nrows=35000)
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 (35000, 17)
--------------------------------------------------
the attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

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

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

Out[4]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00

	id	description	quantity	price
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [5]:

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

Out[5]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [6]:

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

In [7]:

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

Out[7]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade

In [8]:

```
# presence of the numerical digits in a strings with numeric :
https://stackoverflow.com/a/19859308/8089731

def hasNumbers(inputString):
    return any(i.isdigit()for i in inputString)

p1=project_data[['id','project_resource_summary']]
p1=pd.DataFrame(data=p1)
p1.columns=['id','digits_in_summary']
p1['digits_in_summary']=p1['digits_in_summary'].map(hasNumbers)

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

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

1.2 preprocessing of project_subject_categories

In [9]:

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

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

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

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

In [10]:

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

Out[10]:

```
Counter({'Literacy_Language': 16763,
         'History_Civics': 1891,
         'Health_Sports': 4604,
         'Math_Science': 13186,
         'SpecialNeeds': 4357,
         'AppliedLearning': 3879,
         'Music_Arts': 3280,
         'Warmth': 466,
         'Care_Hunger': 466})
```

In [11]:

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

1.3 preprocessing of project_subject_subcategories

In [12]:

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

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

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

In [13]:

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

Out[13]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade

In [14]:

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

In [15]:

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

1.3 Text preprocessing

In [16]:

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

In [17]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

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

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

In [18]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
```

```
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [19]:

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

```
100%|██████████████████████████████████████████████████████████████████████████| 35000/35000 [01:
44<00:00, 334.96it/s]
```

In [20]:

```
preprocessed_essays[2000]
```

Out[20]:

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

In [21]:

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

```
100%|██████████████████████████████████████████████████████████████████████████| 35000/35000
[00:04<00:00, 7386.71it/s]
```

In [22]:

```
preprocessed_titles[2000]
```

Out[22]:

'steady stools active learning'

In [23]:

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

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

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

```
100%|████████████████████████████████████████████████████████████████| 35000/35000
[00:00<00:00, 68778.38it/s]
```

In [24]:

```
project_grade_cat_list[2000]
```

Out[24]:

'Grades6-8'

In [25]:

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

Out[25]:

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

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

5 rows × 21 columns

In [26]:

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

Out[26]:

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

In [27]:

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

In [28]:

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

In [29]:

```
# School State

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

```
100%|████████████████████████████████████████████████████████████████████████████| 35000/35000
[00:00<00:00, 88062.56it/s]
```

In [30]:

```
project_data['clean_project_school_category'] = project_school_cat_list
project_data.drop(['school_state'], axis=1, inplace=True)
project_data.head(2)
```

Out[30]:

	Unnamed: 0	id	teacher_id	teacher_prefix	project_submitted_datetime	project_title	project_resource_s
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	2016-12-05 13:43:57	Educational Support for English Learners at Home	My stude opportunities to
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	My students need a to help wit

In [31]:

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

```
100%|████████████████████████████████████████████████████████████████████████████| 35000/35000
[00:03<00:00, 10781.88it/s]
```

In [32]:

```
preprocessed_teacher_prefix[20000]
```

Out[32]:

```
'mrs'
```

In [33]:

```
project_data['clean_teacher_prefix_category'] = preprocessed_teacher_prefix
project_data.drop(['teacher_prefix'], axis=1, inplace=True)
project_data.head(2)
```

Out[33]:

	Unnamed: 0	id	teacher_id	project_submitted_datetime	project_title	project_resource_summary	teache
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	2016-12-05 13:43:57	Educational Support for English Learners at Home	My students need opportunities to practice beg...	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	2016-10-25 09:22:10	Wanted: Projector for Hungry Learners	My students need a projector to help with view...	

In [34]:

```
project_data['clean_teacher_prefix_category'][20000]
```

Out[34]:

```
'mrs'
```

1.5 Preparing data for models

2. Random Forest and GBDT

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

In [35]:

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

In [36]:

```
project_data.columns
```

Out[36]:

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

In [37]:

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

In [38]:

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

print(X_train.shape, y_train.shape)
#print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)
```

```
(23450, 19) (23450,)
(11550, 19) (11550,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [39]:

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

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

Vectorizing Categorical data

Response coding

Categories

1. project_subject_categories: clean_categories

In [40]:

```
Xtrain_pos = X_train.loc[X_train['project_is_approved'] == 1]

Xtrain_neg = X_train.loc[X_train['project_is_approved'] == 0]
```

In [41]:

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

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

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

In [42]:

```
pos_cat_p = {}
for p in clean_cat_xtrain.keys():
    pos_cat_p[p] = (clean_pos_cat[p])/float(clean_cat_xtrain[p])


neg_cat_n = {}
for n in clean_cat_xtrain.keys():
    neg_cat_n[n] = (clean_neg_cat[n])/float(clean_cat_xtrain[n])


cat_0n_xtrain = []
cat_1p_xtrain = []
for a in X_train["clean_categories"] :
    b = a.split()
    if len(b) == 1 :
        cat_0n_xtrain.append(neg_cat_n[a])
        cat_1p_xtrain.append(pos_cat_p[a])
    else :
        c = neg_cat_n[b[0]]
        d =neg_cat_n[b[1]]
        e = pos_cat_p[b[0]]
        f = pos_cat_p[b[1]]
        cat_0n_xtrain.append(c*d)
        cat_1p_xtrain.append(e*f)

X_train["cat_0n"] = cat_0n_xtrain

X_train["cat_1p"] = cat_1p_xtrain
```

In [43]:

```
import math

cat_0n_xtest = []
cat_1p_xtest = []
for a in X_test["clean_categories"] :
    b = a.split()
    if len(b) == 1 :
        cat_0n_xtest.append(neg_cat_n[a])
        cat_1p_xtest.append(pos_cat_p[a])
    else :
        c = neg_cat_n[b[0]]
        d = neg_cat_n[b[1]]
        e = pos_cat_p[b[0]]
        f = pos_cat_p[b[1]]
        cat_0n_xtest.append(c*d)
        cat_1p_xtest.append(e*f)


flag = 0
for i in range(len(cat_0n_xtest)):
    if(math.isnan(cat_0n_xtest[i])):
        flag = 1
print(flag)
```

```
0
```

In [44]:

```
X_test["cat_0n"] = cat_0n_xtest

X_test["cat_1p"] = cat_1p_xtest
```

In [45]:

```
#https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler


cat_std0n = StandardScaler()
cat_std0n.fit(X_train['cat_0n'].values.reshape(-1,1)) # finding the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
cat_0n_xtrain = cat_std0n.transform(X_train['cat_0n'].values.reshape(-1, 1))
cat_0n_xtest = cat_std0n.transform(X_test['cat_0n'].values.reshape(-1, 1))
print(cat_0n_xtrain.shape)
print(cat_0n_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

In [46]:

```
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler


cat_std1p = StandardScaler()
cat_std1p.fit(X_train['cat_1p'].values.reshape(-1,1)) # finding the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
cat_1p_xtrain = cat_std1p.transform(X_train['cat_1p'].values.reshape(-1, 1))
cat_1p_xtest = cat_std1p.transform(X_test['cat_1p'].values.reshape(-1, 1))
print(cat_1p_xtrain.shape)
print(cat_1p_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

2. project_subject_subcategories : clean_subcategories

In [47]:

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

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

clean_sub_cat_xtrain = {}
for a in X_train['clean_subcategories'] :
```

```
    for b in a.split():
        if b not in clean_sub_cat_xtrain :
            clean_sub_cat_xtrain[b] = 1
        else :
            clean_sub_cat_xtrain[b] += 1
```

In [48]:

```
pos_sub_cat_p = {}
for p in clean_sub_cat_xtrain.keys():
    pos_sub_cat_p[p] = (clean_pos_sub_cat[p])/float(clean_sub_cat_xtrain[p])


neg_sub_cat_n = {}
for n in clean_sub_cat_xtrain.keys():
    neg_sub_cat_n[n] = (clean_neg_sub_cat[n])/float(clean_sub_cat_xtrain[n])


sub_cat_0n_xtrain = []
sub_cat_1p_xtrain = []
for a in X_train['clean_subcategories'] :
    b = a.split()
    if len(b) == 1 :
        sub_cat_0n_xtrain.append(neg_sub_cat_n[a])
        sub_cat_1p_xtrain.append(pos_sub_cat_p[a])
    else :
        c = neg_sub_cat_n[b[0]]
        d =neg_sub_cat_n[b[1]]
        e = pos_sub_cat_p[b[0]]
        f = pos_sub_cat_p[b[1]]
        sub_cat_0n_xtrain.append(c*d)
        sub_cat_1p_xtrain.append(e*f)

X_train["sub_cat_0n"] = sub_cat_0n_xtrain

X_train["sub_cat_1p"] = sub_cat_1p_xtrain
```

In [49]:

```
import math

sub_cat_0n_xtest = []
sub_cat_1p_xtest = []
for a in X_test['clean_subcategories'] :
    b = a.split()
    if len(b) == 1 :
        sub_cat_0n_xtest.append(neg_sub_cat_n[a])
        sub_cat_1p_xtest.append(pos_sub_cat_p[a])
    else :
        c = neg_sub_cat_n[b[0]]
        d = neg_sub_cat_n[b[1]]
        e = pos_sub_cat_p[b[0]]
        f = pos_sub_cat_p[b[1]]
        sub_cat_0n_xtest.append(c*d)
        sub_cat_1p_xtest.append(e*f)


flag = 0
for i in range(len(sub_cat_0n_xtest)):
    if(math.isnan(sub_cat_0n_xtest[i])):
        flag = 1
print(flag)
```

```
0
```

In [50]:

```
X_test["sub_cat_0n"] = cat_0n_xtest

X_test["sub_cat_1p"] = cat_1p_xtest
```

In [51]:

```
#https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler


sub_cat_std0n = StandardScaler()
sub_cat_std0n.fit(X_train['sub_cat_0n'].values.reshape(-1,1)) # finding the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
sub_cat_0n_xtrain = sub_cat_std0n.transform(X_train['sub_cat_0n'].values.reshape(-1, 1))
sub_cat_0n_xtest = cat_std0n.transform(X_test['sub_cat_0n'].values.reshape(-1, 1))
print(sub_cat_0n_xtrain.shape)
print(sub_cat_0n_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

In [52]:

```
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler


sub_cat_std1p = StandardScaler()
sub_cat_std1p.fit(X_train['sub_cat_1p'].values.reshape(-1,1)) # finding the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
sub_cat_1p_xtrain = sub_cat_std1p.transform(X_train['sub_cat_1p'].values.reshape(-1, 1))
sub_cat_1p_xtest = sub_cat_std1p.transform(X_test['sub_cat_1p'].values.reshape(-1, 1))
print(sub_cat_1p_xtrain.shape)
print(sub_cat_1p_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

3. School State

In [53]:

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

school_state_neg = {}
for a in Xtrain_neg['clean_project_school_category'] :
    if a not in school_state_neg :
        school_state_neg[a] = 1
    else :
        school_state_neg[a] += 1

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

In [54]:

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

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

school_state_0n_xtrain = []
school_state_1p_xtrain = []
for a in X_train['clean_project_school_category'] :
    school_state_0n_xtrain.append(neg_school_state_n[a])
    school_state_1p_xtrain.append(pos_school_state_p[a])
```

In [55]:

```
X_train["school_state_0n"] =school_state_0n_xtrain
X_train["school_state_1p"] = school_state_1p_xtrain
```

In [56]:

```
school_state_0n_xtest = []
school_state_1p_xtest = []
for a in X_test['clean_project_school_category'] :
    school_state_0n_xtest.append(neg_school_state_n[a])
    school_state_1p_xtest.append(pos_school_state_p[a])

flag = 0
for i in range(len(school_state_0n_xtest)):
    if(math.isnan(school_state_0n_xtest[i])):
        flag = 1
print(flag)
```

```
0
```

In [57]:

```
X_test["school_state_0n"] =school_state_0n_xtest

X_test["school_state_1p"] =school_state_1p_xtest
```

In [58]:

```
#https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

school_state_std0 = StandardScaler()
school_state_std0.fit(X_train["school_state_0n"].values.reshape(-1,1)) # finding the mean and standard deviation of this data


school_state_0n_xtrain = school_state_std0.transform(X_train["school_state_0n"].values.reshape(-1, 1))
school_state_0n_xtest = school_state_std0.transform(X_test["school_state_0n"].values.reshape(-1, 1))
print(school_state_0n_xtrain.shape)
print(school_state_0n_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

In [59]:

```
#https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

school_state_std1 = StandardScaler()
school_state_std1.fit(X_train["school_state_1p"].values.reshape(-1,1)) # finding the mean and
```

```
standard deviation of this data


school_state_1p_xtrain = school_state_std1.transform(X_train["school_state_1p"].values.reshape(-1,
1))
school_state_1p_xtest = school_state_std1.transform(X_test["school_state_1p"].values.reshape(-1, 1)
)
print(school_state_1p_xtrain.shape)
print(school_state_1p_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

4. Teacher Prefix

In [60]:

```
teacher_prefix_pos = {}
for a in Xtrain_pos['clean_teacher_prefix_category'] :
    if a not in teacher_prefix_pos :
        teacher_prefix_pos[a] = 1
    else :
        teacher_prefix_pos[a] += 1

teacher_prefix_neg = {}
for a in Xtrain_neg['clean_teacher_prefix_category'] :
    if a not in teacher_prefix_neg :
        teacher_prefix_neg[a] = 1
    else :
        teacher_prefix_neg[a] += 1

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

In [61]:

```
pos_teacher_pref = {}
for p in teacher_prefix_xtrain.keys():
    pos_teacher_pref[p] = (teacher_prefix_pos[p])/float(teacher_prefix_xtrain[p])


neg_teacher_pref = {}
for n in teacher_prefix_xtrain.keys():
    neg_teacher_pref[n] = (teacher_prefix_neg[n])/float(teacher_prefix_xtrain[n])


teacher_pref_0n_xtrain = []
teacher_pref_1p_xtrain = []
for a in X_train['clean_teacher_prefix_category'] :
    teacher_pref_0n_xtrain.append(neg_teacher_pref[a])
    teacher_pref_1p_xtrain.append(pos_teacher_pref[a])
```

In [62]:

```
X_train["teacher_prefix_0n"] = teacher_pref_0n_xtrain

X_train["teacher_prefix_1p"] = teacher_pref_1p_xtrain
```

In [63]:

```
teacher_pref_0n_xtest = []
teacher_pref_1p_xtest = []
```

In [64]:

```
for a in X_test['clean_teacher_prefix_category'] :
```

```
    teacher_pref_0n_xtest.append(neg_teacher_pref[a])
    teacher_pref_1p_xtest.append(pos_teacher_pref[a])
```

In [65]:

```
flag = 0
for i in range(len(teacher_pref_0n_xtest)):
    if(math.isnan(teacher_pref_0n_xtest[i])):
        flag = 1
print(flag)
```

```
0
```

In [66]:

```
X_test["teacher_prefix_0n"] = teacher_pref_0n_xtest

X_test["teacher_prefix_1p"] = teacher_pref_1p_xtest
```

In [67]:

```
#https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler


teacher_pref_std0 = StandardScaler()
teacher_pref_std0.fit(X_train['teacher_prefix_0n'].values.reshape(-1,1)) # finding the mean and standard deviation of this data


teacher_prefix_0n_xtrain = teacher_pref_std0.transform(X_train['teacher_prefix_0n'].values.reshape(-1, 1))
teacher_prefix_0n_xtest = teacher_pref_std0.transform(X_test['teacher_prefix_0n'].values.reshape(-1, 1))
print(teacher_prefix_0n_xtrain.shape)
print(teacher_prefix_0n_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

In [68]:

```
#https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler


teacher_pref_std1  = StandardScaler()
teacher_pref_std1.fit(X_train['teacher_prefix_1p'].values.reshape(-1,1)) # finding the mean and standard deviation of this data


teacher_prefix_1p_xtrain = teacher_pref_std1.transform(X_train['teacher_prefix_1p'].values.reshape(-1, 1))
teacher_prefix_1p_xtest = teacher_pref_std1.transform(X_test['teacher_prefix_1p'].values.reshape(-1, 1))
print(teacher_prefix_1p_xtrain.shape)
print(teacher_prefix_1p_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

5. Project Grade

In [69]:

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


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


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

In [70]:

```
pos_proj_grade = {}
for p in proj_grade_xtrain.keys():
    pos_proj_grade[p] = (proj_grade_pos[p])/float(proj_grade_xtrain[p])

neg_proj_grade = {}
for n in proj_grade_xtrain.keys():
    neg_proj_grade[n] = (proj_grade_neg[n])/float(proj_grade_xtrain[n])

proj_grade_0n_xtrain = []
proj_grade_1p_xtrain = []
for a in X_train["clean_project_grade_category"] :
    proj_grade_0n_xtrain.append(neg_proj_grade[a])
    proj_grade_1p_xtrain.append(pos_proj_grade[a])
```

In [71]:

```
X_train["proj_grade_0n"] = proj_grade_0n_xtrain

X_train["proj_grade_1p"] = proj_grade_1p_xtrain
```

In [72]:

```
proj_grade_0n_xtest = []
proj_grade_1p_xtest = []
for a in X_test["clean_project_grade_category"] :
    proj_grade_0n_xtest.append(neg_proj_grade[a])
    proj_grade_1p_xtest.append(pos_proj_grade[a])

flag = 0
for i in range(len(proj_grade_0n_xtest)):
    if(math.isnan(proj_grade_0n_xtest[i])):
        flag = 1
print(flag)
```

```
0
```

In [73]:

```
X_test["proj_grade_0n"] = proj_grade_0n_xtest

X_test["proj_grade_1p"] = proj_grade_1p_xtest
```

In [74]:

```
#https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler


proj_grade_std0 = StandardScaler()
proj_grade_std0.fit(X_train['proj_grade_0n'].values.reshape(-1,1)) # finding the mean and standard deviation of this data


proj_grade_0n_xtrain = proj_grade_std0 .transform(X_train['proj_grade_0n'].values.reshape(-1, 1))
proj_grade_0n_xtest = proj_grade_std0 .transform(X_test['proj_grade_0n'].values.reshape(-1, 1))
print(proj_grade_0n_xtrain.shape)
print(proj_grade_0n_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

In [76]:

```
#https://www.analyticsvidhya.com/blog/2015/11/easy-methods-deal-categorical-variables-predictive-modeling/
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler


proj_grade_std1 = StandardScaler()
proj_grade_std1.fit(X_train['proj_grade_1p'].values.reshape(-1,1)) # finding the mean and standard deviation of this data


proj_grade_1p_xtrain = proj_grade_std1.transform(X_train['proj_grade_1p'].values.reshape(-1, 1))
proj_grade_1p_xtest = proj_grade_std1.transform(X_test['proj_grade_1p'].values.reshape(-1, 1))
print(proj_grade_1p_xtrain.shape)
print(proj_grade_1p_xtest.shape)
```

```
(23450, 1)
(11550, 1)
```

Vectorizing Numerical features

In [77]:

```
# 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_xtrain = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
#price_standardized_xcv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_standardized_xtest = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
print("shape of price_standardized_xtrain",price_standardized_xtrain.shape)
#print("shape of price_standardized_xcv",price_standardized_xcv.shape)
print("shape of price_standardized_xtest",price_standardized_xtest.shape)
```

```
shape of price_standardized_xtrain (23450, 1)
```

```
shape of price_standardized_xtest (11550, 1)
```

In [78]:

```
# 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)

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

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

```
C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

```
shape of quantity_standardized_xtrain (23450, 1)
shape of quantity_standardized_xtest (11550, 1)
```

In [79]:

```
# 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)

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

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

```
teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
print(" shape of
teacher_number_of_previously_posted_projects_standardized_xtrain",teacher_num_prev_projects_standar
ized_xtrain.shape)
#print(" shape of
teacher_number_of_previously_posted_projects_standardized_xcv",teacher_num_prev_projects_standardi
cv.shape)
print(" shape of
teacher_number_of_previously_posted_projects_standardized_xtest",teacher_num_prev_projects_standar
zed_xtest.shape)
```

```
C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\Santosh\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

```
 shape of teacher_number_of_previously_posted_projects_standardized_xtrain (23450, 1)
 shape of teacher_number_of_previously_posted_projects_standardized_xtest (11550, 1)
```

Vectorizing Text data

BOW on eassay

2.3 Make Data Model Ready: encoding eassay, and project_title

In [ ]:

```
# 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
```

In [80]:

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

vectorizer_bow_essays = CountVectorizer(min_df=10,max_features=5000,ngram_range=(1,2))
vectorizer_bow_essays.fit(X_train['preprocessed_essays'])

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

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

```
Shape of matrix after BOW_text_essay X_train  (23450, 5000)
Shape of matrix after BOW_text_essay X_test  (11550, 5000)
```

BOW on project_title

In [81]:

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

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

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

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

```
Shape of matrix after BOW project_title_xtrain  (23450, 1294)
Shape of matrix after BOW project_title_xtest  (11550, 1294)
```

TFIDF Vectorizer on Essay

In [82]:

```
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'])

essay_tfidf_xtrain = vectorizer_tfidf_essays.transform(X_train['preprocessed_essays'])
#essay_tfidf_xcv = vectorizer_tfidf_essays.transform(X_cv['preprocessed_essays'])
essay_tfidf_xtest = vectorizer_tfidf_essays.transform(X_test['preprocessed_essays'])

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

```
Shape of matrix after tfidf eassay_xtrain  (23450, 5000)
Shape of matrix after tfidf essay_xtest  (11550, 5000)
```

TFIDF Vectorizer on Project Title

In [83]:

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

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

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

```
Shape of matrix after tfidf proj_title_xtrain  (23450, 1294)
Shape of matrix after tfidf proj_title_xtest  (11550, 1294)
```

In [84]:

```
# Using Pretrained Models: Avg W2V
```

In [85]:

In [85]:

```
# 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())
```

Average Word2Vec on Essay

In [86]:

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

# average Word2Vec on X_train
essay_avg_w2v_vectors_xtrain = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['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
    essay_avg_w2v_vectors_xtrain.append(vector)

print(len(essay_avg_w2v_vectors_xtrain))
print(len(essay_avg_w2v_vectors_xtrain[0]))



# average Word2Vec on X_test

essay_avg_w2v_vectors_xtest = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['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
    essay_avg_w2v_vectors_xtest.append(vector)

print(len(essay_avg_w2v_vectors_xtest))
print(len(essay_avg_w2v_vectors_xtest[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 23450/23450 [00:34<00:00, 686.74it/s]
```

```
23450
300
```

```
100%|██████████████████████████████████████████████████████████████████████████| 11550/11550 [00:15<00:00, 746.09it/s]
```

```
11550
300
```

Average Word2Vec on Project Title

In [87]:

```
# average Word2Vec

# compute average word2vec for each review
```

```
# Compute average Word2Vec for each review.

# average Word2Vec on X_train

proj_title_avg_w2v_vectors_xtrain = []; # the avg-w2v for each sentence/review is stored in this l
ist
for sentence in tqdm(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
    proj_title_avg_w2v_vectors_xtrain.append(vector)

print(len(proj_title_avg_w2v_vectors_xtrain))
print(len(proj_title_avg_w2v_vectors_xtrain[0]))


# average Word2Vec on X_test

proj_title_avg_w2v_vectors_xtest = []; # the avg-w2v for each sentence/review is stored in this li
st
for sentence in tqdm(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
    proj_title_avg_w2v_vectors_xtest.append(vector)

print(len(proj_title_avg_w2v_vectors_xtest))
print(len(proj_title_avg_w2v_vectors_xtest[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 23450/23450
[00:01<00:00, 12614.55it/s]
```

```
23450
300
```

```
100%|██████████████████████████████████████████████████████████████████████████| 11550/11550
[00:00<00:00, 14471.49it/s]
```

```
11550
300
```

Using Pretrained Models: TFIDF weighted W2V

TFIDF weighted W2V on Essays

In [88]:

```
# 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())
```

In [89]:

```
# average Word2Vec

# average Word2Vec on X_train
```

```
essay_tfidf_w2v_vectors_xtrain = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['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 tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_vectors_xtrain.append(vector)

print(len(essay_tfidf_w2v_vectors_xtrain))
print(len(essay_tfidf_w2v_vectors_xtrain[0]))



# average Word2Vec on X_train
essay_tfidf_w2v_vectors_xtest = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(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 tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_vectors_xtest.append(vector)

print(len(essay_tfidf_w2v_vectors_xtest))
print(len(essay_tfidf_w2v_vectors_xtest[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████| 23450/23450 [04
:39<00:00, 84.04it/s]
```

```
23450
300
```

```
100%|████████████████████████████████████████████████████████████████████████| 11550/11550 [02
:11<00:00, 88.14it/s]
```

```
11550
300
```

TFIDF weighted W2V on Project Title

In [90]:

```
# 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())
```

In [91]:

```
# TFIDF weighted W2V on Project Title
# compute average word2vec for each review.
```

```
# TFIDF weighted W2V on X_train

proj_title_tfidf_w2v_vectors_xtrain = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(X_train['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    proj_title_tfidf_w2v_vectors_xtrain.append(vector)

print(len(proj_title_tfidf_w2v_vectors_xtrain))
print(len(proj_title_tfidf_w2v_vectors_xtrain[0]))



# TFIDF weighted W2V on X_test
proj_title_tfidf_w2v_vectors_xtest = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(X_test['preprocessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    proj_title_tfidf_w2v_vectors_xtest.append(vector)

print(len(proj_title_tfidf_w2v_vectors_xtest))
print(len(proj_title_tfidf_w2v_vectors_xtest[0]))
```

```
100%|████████████████████████████████████████████████████████████████| 23450/23450
[00:03<00:00, 6109.23it/s]
```

```
23450
300
```

```
100%|████████████████████████████████████████████████████████████████| 11550/11550
[00:01<00:00, 6507.91it/s]
```

```
11550
300
```

2.4 Applying Random Forest

In [92]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
```

```
    # c. X-axis label
    # d. Y-axis label
```

2.4.1 Applying Random Forest on BOW, SET 1

In [93]:

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

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


X_train1=hstack((cat_0n_xtrain, cat_1p_xtrain, sub_cat_0n_xtrain, sub_cat_1p_xtrain,
school_state_0n_xtrain,
                 school_state_1p_xtrain, teacher_prefix_0n_xtrain, teacher_prefix_1p_xtrain,
                 proj_grade_0n_xtrain, proj_grade_1p_xtrain, price_standardized_xtrain,
                teacher_num_prev_projects_standardized_xtrain, quantity_standardized_xtrain,
                 essay_text_bow_xtrain, proj_title_bow_xtrain)).tocsr()



X_test1=hstack((cat_0n_xtest, cat_1p_xtest, sub_cat_0n_xtest, sub_cat_1p_xtest,
school_state_0n_xtest,
                 school_state_1p_xtest, teacher_prefix_0n_xtest, teacher_prefix_1p_xtest,
                 proj_grade_0n_xtest, proj_grade_1p_xtest, price_standardized_xtest,
                teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                 essay_text_bow_xtest, proj_title_bow_xtest)).tocsr()

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

```
(23450, 6307) (23450,)
(11550, 6307) (11550,)
```

GridSearchCV

In [94]:

```
import warnings
warnings.filterwarnings('ignore')
import time

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

start_time = time.time()

rfclf1 = RandomForestClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 300, 500, 700], 'max_depth':[10, 30, 60, 100]}
clf1 = GridSearchCV(rfclf1, parameters, cv= 5, scoring='roc_auc',return_train_score=True)
clf1.fit(X_train1, y_train)

train_auc= clf1.cv_results_['mean_train_score']
train_auc_std= clf1.cv_results_['std_train_score']
cv_auc = clf1.cv_results_['mean_test_score']
cv_auc_std= clf1.cv_results_['std_test_score']


print("Total Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Total Execution time: 10894.524728775024 ms
```

In [ ]:

```
## 8.15-11.17=3 hours
```

In [95]:

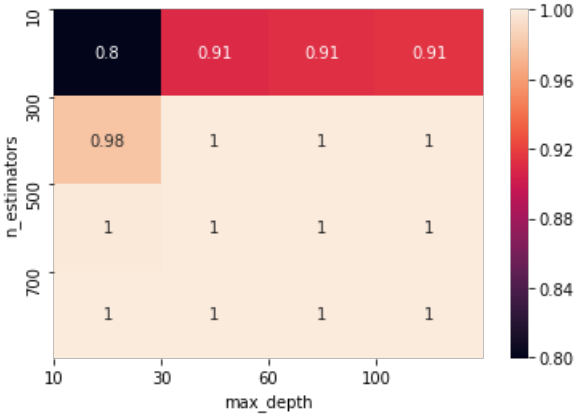
```
train_auc = train_auc.reshape(4,4)

train_auc
```

Out[95]:

```
array([[0.79961755, 0.90929167, 0.91266686, 0.91440683],
       [0.97928145, 0.99986802, 0.99992368, 0.99994144],
       [0.99818957, 1.        , 1.        , 1.        ],
       [0.99952739, 1.        , 1.        , 1.        ]])
```

In [96]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(train_auc,annot=True)

plt.xticks(np.arange(4), [10, 30, 60, 100])
plt.yticks(np.arange(4), [10, 300, 500, 700])


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


plt.show()
```



In [97]:

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

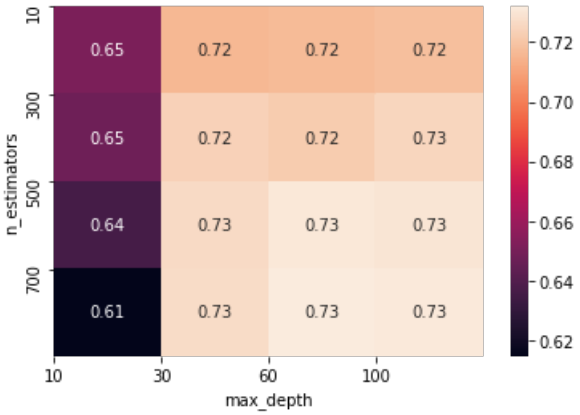
Out[97]:

```
array([[0.65153356, 0.71615098, 0.71690264, 0.71848891],
       [0.64794705, 0.72361836, 0.72207499, 0.72510461],
       [0.63623846, 0.72667368, 0.73072985, 0.72988267],
       [0.61477431, 0.72709737, 0.73177981, 0.73127221]])
```

In [98]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(cv_auc,annot=True)

plt.xticks(np.arange(4), [10, 30, 60, 100])
plt.yticks(np.arange(4), [10, 300, 500, 700])
```

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


plt.show()
```



Conclusion

Choosing max_depth=10, n_estimators=500 as best parameters

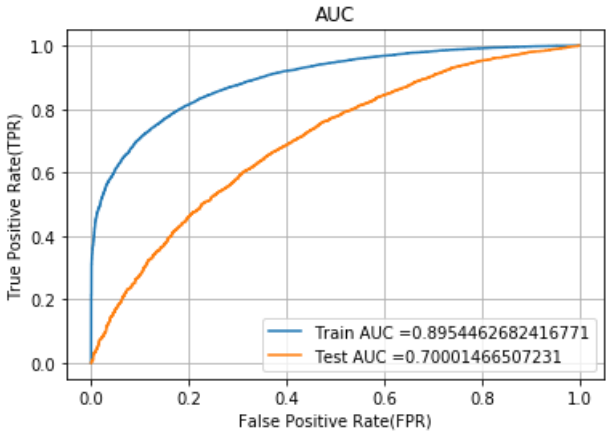
Simple for loop (if you are having memory limitations use this)

In [100]:

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

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

In [107]:

```
from sklearn.metrics import roc_curve, auc

modelbow = RandomForestClassifier(max_depth = 10, n_estimators = 500,n_jobs=-1,class_weight='balanc
ed')
modelbow.fit(X_train1, y_train)


y_train_pred = batch_predict(modelbow, X_train1)
y_test_pred = batch_predict(modelbow, X_test1)

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(FPR)")
plt.ylabel("True Positive Rate(TPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [108]:

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

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

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

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

In [109]:

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

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999808272225 for threshold 0.476
[[ 1806  1805]
 [ 1037 18802]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.498
[[ 817  961]
 [1924 7848]]
```

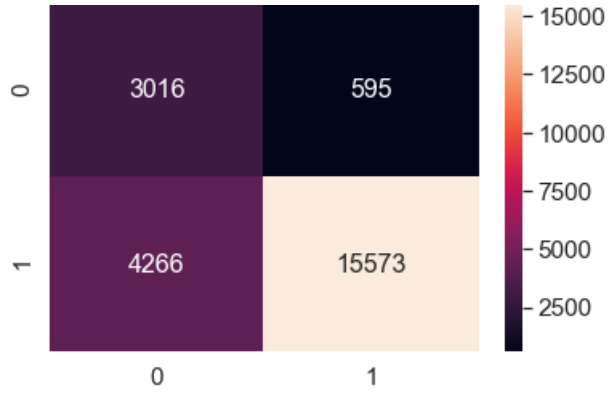
In [110]:

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

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

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

```
the maximum value of tpr*(1-fpr) 0.6556262822952683 for threshold 0.506
```

Out[110]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b5599c8c50>
```



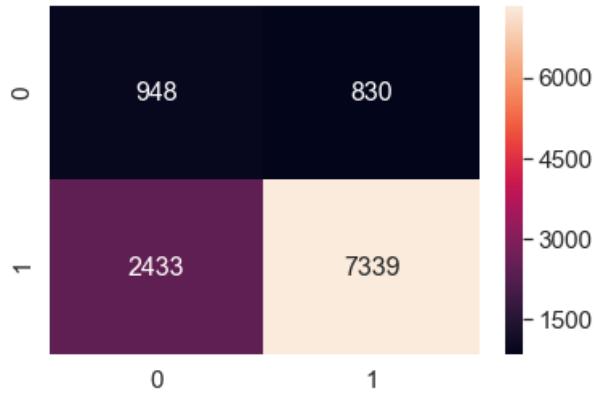
In [111]:

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

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

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

```
the maximum value of tpr*(1-fpr) 0.41827934499386915 for threshold 0.505
```

Out[111]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b556d3d710>
```



In [1]:

```
import dill
#dill.dump_session('notebook_4_11.db')
dill.load_session('notebook_4_11.db')
```

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

2.4.2 Applying Logistic Regression on TFIDF, SET 2

In [2]:

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

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

X_train2=hstack((cat_0n_xtrain, cat_1p_xtrain, sub_cat_0n_xtrain, sub_cat_1p_xtrain,
school_state_0n_xtrain,
```

```
                school_state_1p_xtrain, teacher_prefix_0n_xtrain, teacher_prefix_1p_xtrain,
                proj_grade_0n_xtrain, proj_grade_1p_xtrain, price_standardized_xtrain,
                teacher_num_prev_projects_standardized_xtrain,
                quantity_standardized_xtrain,essay_tfidf_xtrain, proj_title_tfidf_xtrain)).tocsr()



X_test2=hstack((cat_0n_xtest, cat_1p_xtest, sub_cat_0n_xtest, sub_cat_1p_xtest,
school_state_0n_xtest,
                school_state_1p_xtest, teacher_prefix_0n_xtest, teacher_prefix_1p_xtest,
                proj_grade_0n_xtest, proj_grade_1p_xtest, price_standardized_xtest,
               teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                essay_tfidf_xtest, proj_title_tfidf_xtest)).tocsr()


print(X_train2.shape)

print(X_test2.shape)
```

```
(23450, 6307)
(11550, 6307)
```

GridSearchCV

In [3]:

```
import warnings
warnings.filterwarnings('ignore')
import time

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

start_time = time.time()

rfclf2 = RandomForestClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 300, 500, 700], 'max_depth':[10, 30, 60, 100]}
clf2 = GridSearchCV(rfclf2, parameters, cv= 5, scoring='roc_auc',return_train_score=True)
clf2.fit(X_train2, y_train)

train_auc= clf2.cv_results_['mean_train_score']
train_auc_std= clf2.cv_results_['std_train_score']
cv_auc = clf2.cv_results_['mean_test_score']
cv_auc_std= clf2.cv_results_['std_test_score']


print("Total Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Total Execution time: 13887.828559160233 ms
```

In [1]:

```
import dill
#dill.dump_session('notebook_44_11.db')
dill.load_session('notebook_44_11.db')
```

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

In [2]:

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

train_auc
```

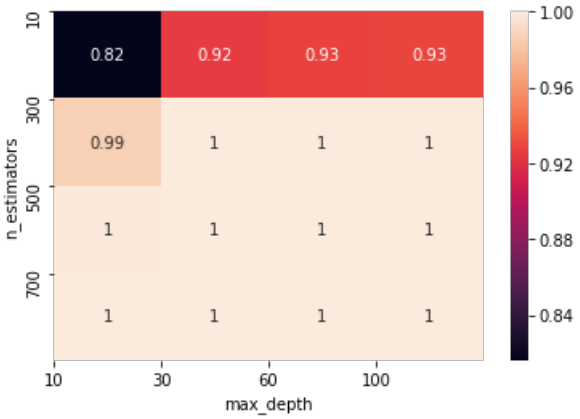
Out[2]:

```
array([[0.81607497, 0.92464136, 0.92780219, 0.92822427],
```

```
       [0.98690655, 0.99999668, 0.9999987 , 0.99999898],
       [0.99926093, 1.        , 1.        , 1.        ],
       [0.99965037, 1.        , 1.        , 1.        ]])
```

In [3]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(train_auc,annot=True)

plt.xticks(np.arange(4), [10, 30, 60, 100])
plt.yticks(np.arange(4), [10, 300, 500, 700])


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


plt.show()
```



In [4]:

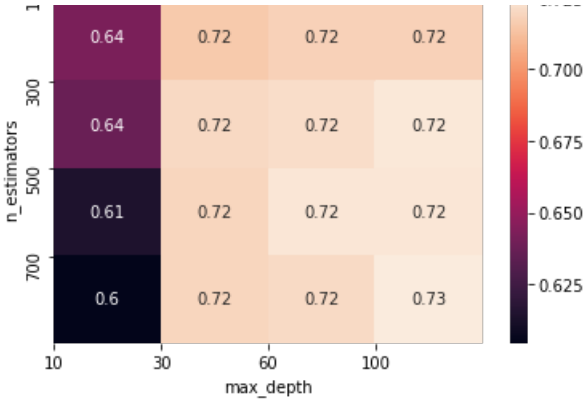
```
cv_auc =cv_auc.reshape(4,4)
cv_auc
```

Out[4]:

```
array([[0.64240054, 0.71584298, 0.71786545, 0.71808724],
       [0.63767698, 0.72012996, 0.72178875, 0.72499285],
       [0.61427333, 0.71944671, 0.72479341, 0.72462129],
       [0.60456107, 0.71936093, 0.72091629, 0.72630838]])
```

In [5]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(cv_auc,annot=True)

plt.xticks(np.arange(4), [10, 30, 60, 100])
plt.yticks(np.arange(4), [10, 300, 500, 700])

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


plt.show()
```



Conclusion

Choosing max_depth=10, n_estimators=100 as best parameters

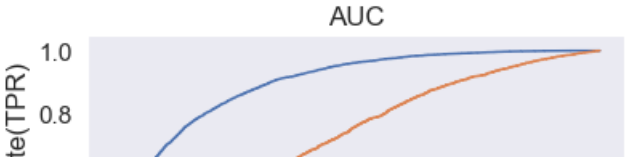
Simple for loop (if you are having memory limitations use this)

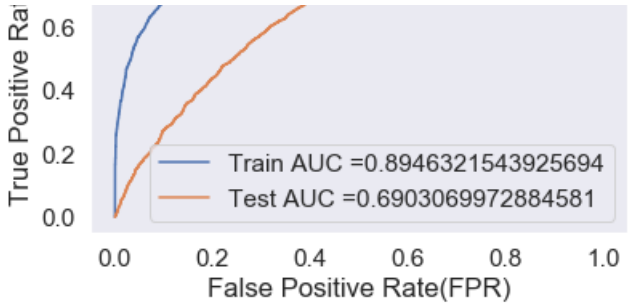
In [7]:

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

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

In [16]:

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

modeltfidf = RandomForestClassifier(max_depth = 10, n_estimators = 100,n_jobs=-1,class_weight='bala
nced')
modeltfidf.fit(X_train2, y_train)


y_train_pred = batch_predict(modeltfidf, X_train2)
y_test_pred = batch_predict(modeltfidf, X_test2)

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(FPR)")
plt.ylabel("True Positive Rate(TPR)")
plt.title("AUC")
plt.grid()
plt.show()
```





In [17]:

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

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

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

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

In [18]:

```
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) 0.6497300080938178 for threshold 0.511
[[ 2963   648]
 [ 4130 15709]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.412533030945835 for threshold 0.513
[[ 991  787]
 [2806 6966]]
```

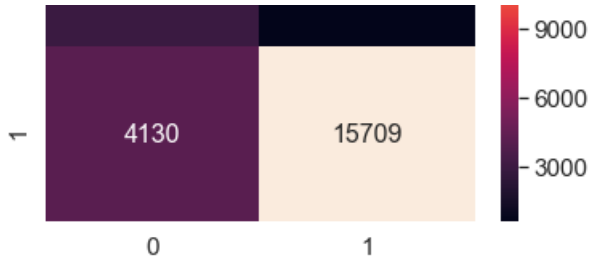
In [19]:

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

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

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

```
the maximum value of tpr*(1-fpr) 0.6497300080938178 for threshold 0.511
```

Out[19]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0d498c1d0>
```



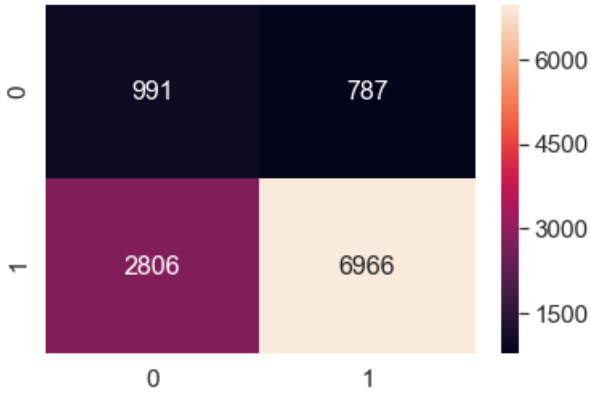
In [20]:

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

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

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

```
the maximum value of tpr*(1-fpr) 0.412533030945835 for threshold 0.513
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f068f588>
```



2.4.1 Applying Logistic Regression on AVG W2V, SET 3

In [22]:

```
essay_avg_w2v_vectors_xtrain= np.array(essay_avg_w2v_vectors_xtrain)
proj_title_avg_w2v_vectors_xtrain= np.array(proj_title_avg_w2v_vectors_xtrain)
```

In [23]:

```
essay_avg_w2v_vectors_xtest= np.array(essay_avg_w2v_vectors_xtest)
proj_title_avg_w2v_vectors_xtest= np.array(proj_title_avg_w2v_vectors_xtest)
```

In [26]:

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

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


X_train3=np.hstack((cat_0n_xtrain, cat_1p_xtrain, sub_cat_0n_xtrain, sub_cat_1p_xtrain,
school_state_0n_xtrain,
                    school_state_1p_xtrain, teacher_prefix_0n_xtrain, teacher_prefix_1p_xtrain,
                    proj_grade_0n_xtrain, proj_grade_1p_xtrain, price_standardized_xtrain,
                teacher_num_prev_projects_standardized_xtrain,
```

```
                essay_avg_w2v_vectors_xtrain, proj_title_avg_w2v_vectors_xtrain))


X_test3=np.hstack((cat_0n_xtest, cat_1p_xtest, sub_cat_0n_xtest, sub_cat_1p_xtest,
school_state_0n_xtest,
                 school_state_1p_xtest, teacher_prefix_0n_xtest, teacher_prefix_1p_xtest,
                 proj_grade_0n_xtest, proj_grade_1p_xtest, price_standardized_xtest,
                teacher_num_prev_projects_standardized_xtest,
                 essay_avg_w2v_vectors_xtest, proj_title_avg_w2v_vectors_xtest))


print(X_train3.shape, y_train.shape)
print(X_test3.shape, y_test.shape)
```

```
(23450, 612) (23450,)
(11550, 612) (11550,)
```

GridSearchCV

In [27]:

```
import warnings
warnings.filterwarnings('ignore')
import time

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

start_time = time.time()

rfclf3 = RandomForestClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 300, 500, 700], 'max_depth':[10, 30, 60, 100]}
clf3 = GridSearchCV(rfclf3, parameters, cv= 5, scoring='roc_auc',return_train_score=True)
clf3.fit(X_train3, y_train)

train_auc= clf3.cv_results_['mean_train_score']
train_auc_std= clf3.cv_results_['std_train_score']
cv_auc = clf3.cv_results_['mean_test_score']
cv_auc_std= clf3.cv_results_['std_test_score']


print("Total Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Total Execution time: 24982.0344414711 ms
```

In [ ]:

```
# 9.53am-5.28pm
```

In [30]:

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

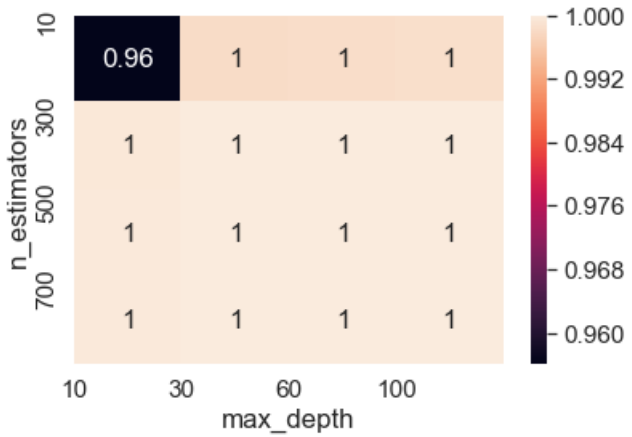
Out[30]:

```
array([[0.95605988, 0.99821308, 0.9984497 , 0.9985575 ],
       [0.99961317, 1.        , 1.        , 1.        ],
       [0.99969757, 1.        , 1.        , 1.        ],
       [0.99969364, 1.        , 1.        , 1.        ]])
```

In [31]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(train_auc,annot=True)
```

```
plt.xticks(np.arange(4), [10, 30, 60, 100])
plt.yticks(np.arange(4), [10, 300, 500, 700])


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


plt.show()
```



In [32]:

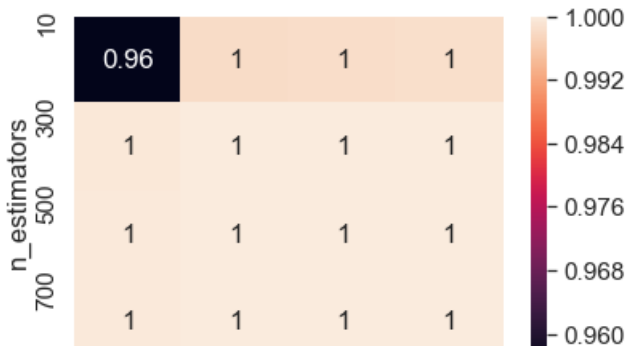
```
cv_auc = train_auc.reshape(4,4)
cv_auc
```

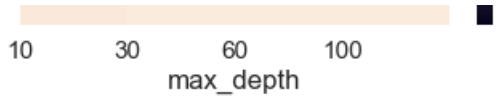
Out[32]:

```
array([[0.95605988, 0.99821308, 0.9984497 , 0.9985575 ],
       [0.99961317, 1.        , 1.        , 1.        ],
       [0.99969757, 1.        , 1.        , 1.        ],
       [0.99969364, 1.        , 1.        , 1.        ]])
```

In [33]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(cv_auc,annot=True)

plt.xticks(np.arange(4), [10, 30, 60, 100])
plt.yticks(np.arange(4), [10, 300, 500, 700])


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


plt.show()
```





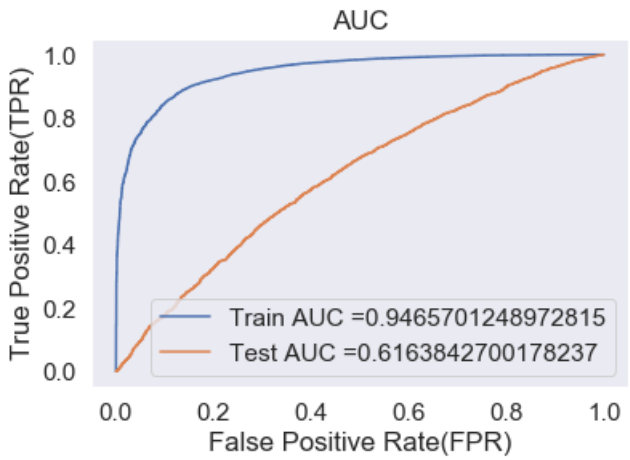
Simple for loop (if you are having memory limitations use this)

In [34]:

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

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

In [44]:

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

modelavgw2v = RandomForestClassifier(max_depth = 10, n_estimators = 10,n_jobs=-1,class_weight='bala
nced')
modelavgw2v.fit(X_train3, y_train)


y_train_pred = batch_predict(modelavgw2v, X_train3)
y_test_pred = batch_predict(modelavgw2v, X_test3)

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(FPR)")
plt.ylabel("True Positive Rate(TPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [45]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
```

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

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

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

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

In [46]:

```
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) 0.7668567749949542 for threshold 0.539
[[ 3158   453]
 [ 2443 17396]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3457749512276991 for threshold 0.505
[[ 411 1367]
 [1221 8551]]
```

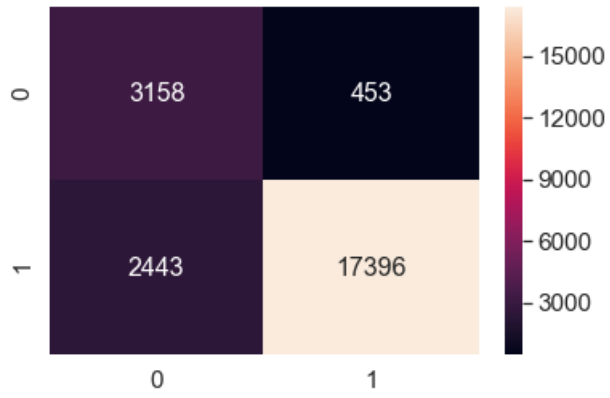
In [47]:

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

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

```
the maximum value of tpr*(1-fpr) 0.7668567749949542 for threshold 0.539
```

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f0a91da0>
```



In [48]:

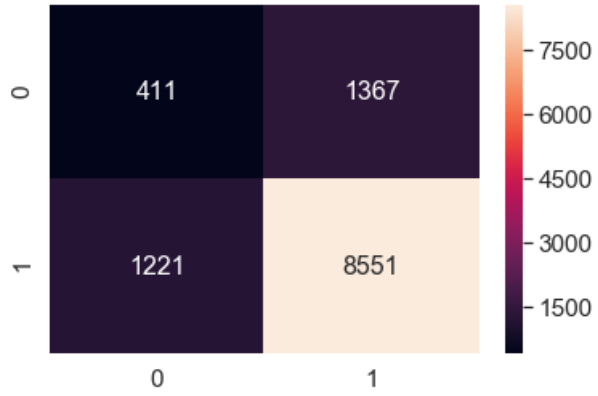
```
# Confusion matrix for test data
# Code for this segment from here -->> https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

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

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

```
the maximum value of tpr*(1-fpr) 0.3457749512276991 for threshold 0.505
```

Out[48]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f098e320>
```



2.4.1 Applying Logistic Regression on TFIDF Word2Vec, SET 4

In [49]:

```
essay_tfidf_w2v_vectors_xtrain=np.array(essay_tfidf_w2v_vectors_xtrain)
proj_title_tfidf_w2v_vectors_xtrain=np.array(proj_title_tfidf_w2v_vectors_xtrain)
```

In [50]:

```
essay_tfidf_w2v_vectors_xtest=np.array(essay_tfidf_w2v_vectors_xtest)
proj_title_tfidf_w2v_vectors_xtest=np.array(proj_title_tfidf_w2v_vectors_xtest)
```

In [51]:

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

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


X_train4=np.hstack((cat_0n_xtrain, cat_1p_xtrain, sub_cat_0n_xtrain, sub_cat_1p_xtrain,
school_state_0n_xtrain,
                 school_state_1p_xtrain, teacher_prefix_0n_xtrain, teacher_prefix_1p_xtrain,
                 proj_grade_0n_xtrain, proj_grade_1p_xtrain, price_standardized_xtrain,
                teacher_num_prev_projects_standardized_xtrain, quantity_standardized_xtrain,
                 essay_tfidf_w2v_vectors_xtrain, proj_title_tfidf_w2v_vectors_xtrain))



X_test4=np.hstack((cat_0n_xtest, cat_1p_xtest, sub_cat_0n_xtest, sub_cat_1p_xtest,
school_state_0n_xtest,
                 school_state_1p_xtest, teacher_prefix_0n_xtest, teacher_prefix_1p_xtest,
                 proj_grade_0n_xtest, proj_grade_1p_xtest, price_standardized_xtest,
                teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                 essay_tfidf_w2v_vectors_xtest, proj_title_tfidf_w2v_vectors_xtest))


print(X_train4.shape, y_train.shape)

print(X_test4.shape, y_test.shape)
```

```
(23450, 613) (23450,)
(11550, 613) (11550,)
```

GridSearchCV

In [52]:

```
import warnings
warnings.filterwarnings('ignore')
import time

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

start_time = time.time()

rfclf4 = RandomForestClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [5, 10, 50, 100], 'max_depth':[2,  5, 7, 10]}
clf4 = GridSearchCV(rfclf4, parameters, cv= 5, scoring='roc_auc',return_train_score=True)
clf4.fit(X_train4, y_train)

train_auc= clf4.cv_results_['mean_train_score']
train_auc_std= clf4.cv_results_['std_train_score']
cv_auc = clf4.cv_results_['mean_test_score']
cv_auc_std= clf4.cv_results_['std_test_score']


print("Total Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Total Execution time: 1334.0021982192993 ms
```

In [ ]:

```
#6.36pm-7.05
```

In [53]:

```
# Testing the performance of the model on test data, plotting ROC Curves
# Select best log(C) value
best_set_tfidfw2v = clf4.best_params_
print(best_set_tfidfw2v)
```

```
{'max_depth': 7, 'n_estimators': 100}
```

In [54]:

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

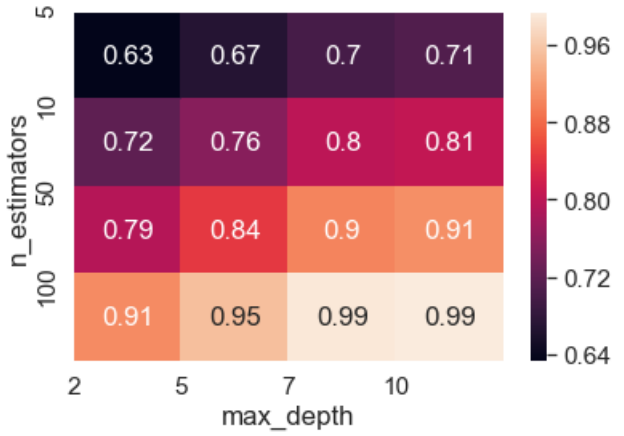
Out[54]:

```
array([[0.63270091, 0.66853872, 0.70013678, 0.70919025],
       [0.72091841, 0.75604252, 0.79957423, 0.80552529],
       [0.79416171, 0.8443032 , 0.90131641, 0.91114463],
       [0.90738306, 0.95332639, 0.99161055, 0.99454705]])
```

In [55]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(train_auc,annot=True)

plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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

```
plt.show()
```



In [56]:

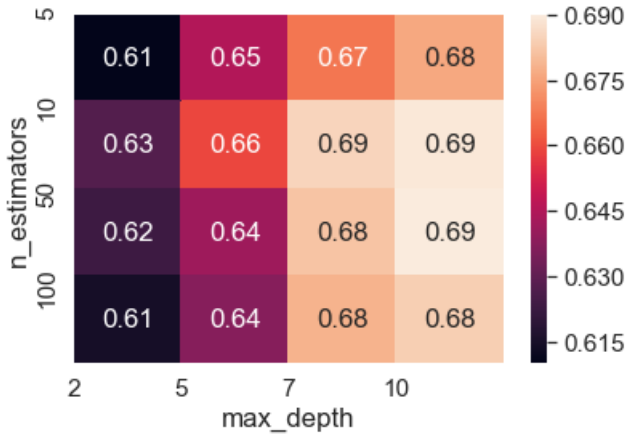
```
cv_auc = cv_auc.reshape(4,4)
cv_auc
```

Out[56]:

```
array([[0.61011862, 0.64533892, 0.66723611, 0.6762549 ],
       [0.6276052 , 0.65927135, 0.68516991, 0.68927965],
       [0.62295525, 0.64148773, 0.68190075, 0.69006752],
       [0.6145868 , 0.63721003, 0.67841932, 0.68391907]])
```

In [57]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(cv_auc,annot=True)

plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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


plt.show()
```

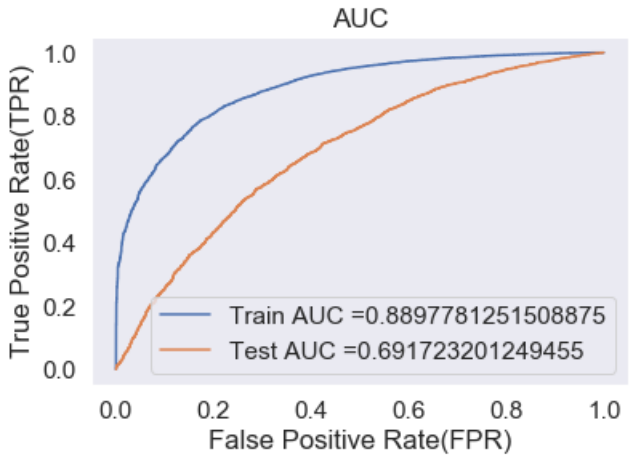


In [58]:

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

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

In [59]:

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

modeltfidfw2v = RandomForestClassifier(max_depth = 7, n_estimators = 100,n_jobs=-1,class_weight='ba
lanced')
modeltfidfw2v.fit(X_train4, y_train)


y_train_pred = batch_predict(modeltfidfw2v, X_train4)
y_test_pred = batch_predict(modeltfidfw2v, X_test4)

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(FPR)")
plt.ylabel("True Positive Rate(TPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [60]:

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

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

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

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

```
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [61]:

```
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) 0.6498779869168071 for threshold 0.515
[[ 2984   627]
 [ 4237 15602]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.41339589893670164 for threshold 0.518
[[ 966  812]
 [2618 7154]]
```

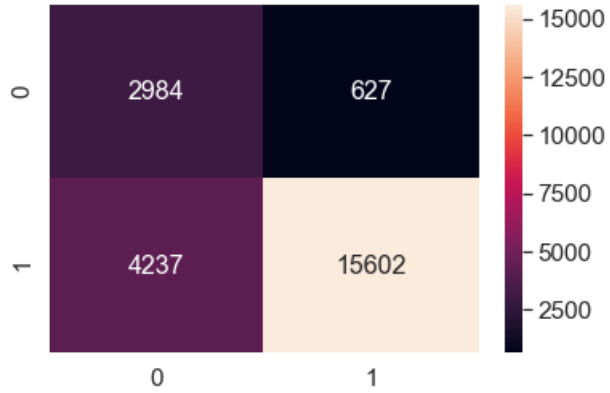
In [62]:

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

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

```
the maximum value of tpr*(1-fpr) 0.6498779869168071 for threshold 0.515
```

Out[62]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f0b4d048>
```



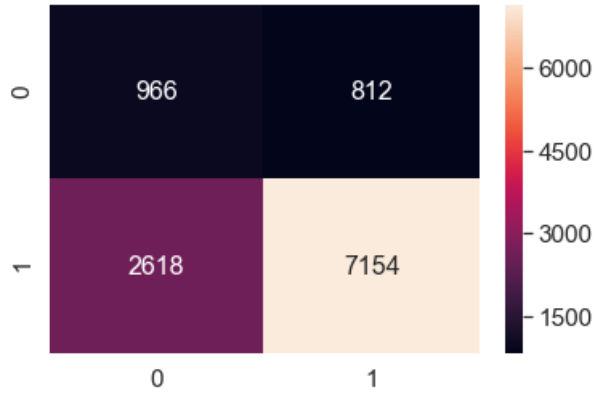
In [63]:

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

conf_matrix_xtest =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matrix_xtest, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.41339589893670164 for threshold 0.518
```

Out[63]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f1ab75c0>
```



In [ ]:

```
```

2.5 Applying GBDT

2.5.1 Applying XGBOOST on BOW, SET 1

In [64]:

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

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


X_train1=hstack((cat_0n_xtrain, cat_1p_xtrain, sub_cat_0n_xtrain, sub_cat_1p_xtrain,
school_state_0n_xtrain,
                 school_state_1p_xtrain, teacher_prefix_0n_xtrain, teacher_prefix_1p_xtrain,
                 proj_grade_0n_xtrain, proj_grade_1p_xtrain, price_standardized_xtrain,
                teacher_num_prev_projects_standardized_xtrain, quantity_standardized_xtrain,
                 essay_text_bow_xtrain, proj_title_bow_xtrain)).tocsr()



X_test1=hstack((cat_0n_xtest, cat_1p_xtest, sub_cat_0n_xtest, sub_cat_1p_xtest,
school_state_0n_xtest,
                 school_state_1p_xtest, teacher_prefix_0n_xtest, teacher_prefix_1p_xtest,
                 proj_grade_0n_xtest, proj_grade_1p_xtest, price_standardized_xtest,
                teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                 essay_text_bow_xtest, proj_title_bow_xtest)).tocsr()

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

```
(23450, 6307) (23450,)
(11550, 6307) (11550,)
```

In [66]:

```
import dill
dill.dump_session('notebook_6_11.db')
#dill.load_session('notebook_env11.db')
```

GridSearchCV

In [67]:

```
from sklearn.model_selection import GridSearchCV
```

```
import xgboost as xgb
import time


start_time = time.time()

gbdt1 = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [5, 10, 50, 100], 'max_depth':[2,  5, 7, 10]}
clfgbdt1 = GridSearchCV(gbdt1, parameters, cv= 3, scoring='roc_auc',return_train_score=True)
clfgbdt1.fit(X_train1, y_train)

train_auc= clfgbdt1.cv_results_['mean_train_score']
train_auc_std= clfgbdt1.cv_results_['std_train_score']
cv_auc = clfgbdt1.cv_results_['mean_test_score']
cv_auc_std= clfgbdt1.cv_results_['std_test_score']

print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 1304.5655992031097 ms
```

In [69]:

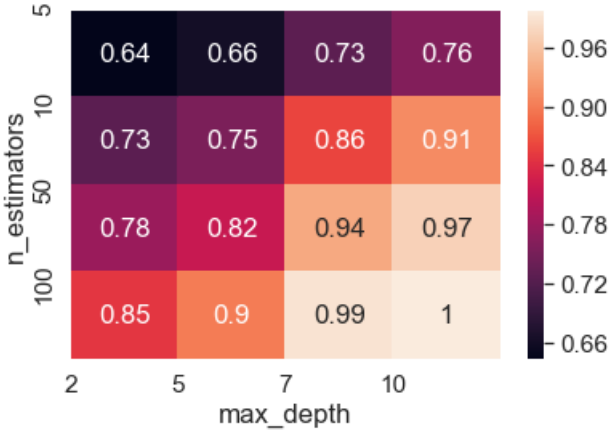
```
train_auc = train_auc.reshape(4,4)

train_auc
```

Out[69]:

```
array([[0.64312452, 0.66162889, 0.72793036, 0.76093421],
       [0.72741995, 0.7541433 , 0.86015555, 0.91342077],
       [0.77955307, 0.81772774, 0.93843885, 0.9740132 ],
       [0.84892386, 0.9000029 , 0.98991558, 0.9981573 ]])
```

In [70]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(train_auc,annot=True)

plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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


plt.show()
```



In [72]:

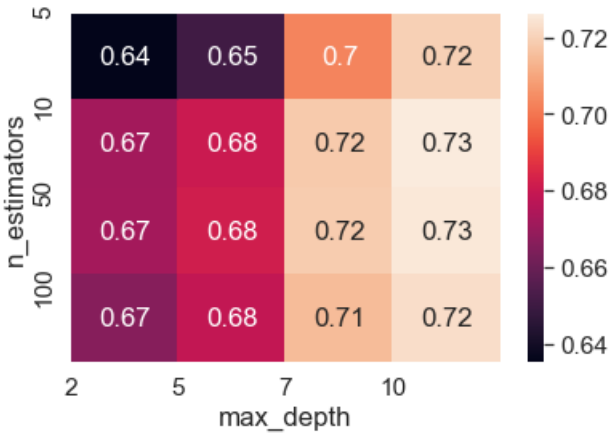
```
cv_auc = cv_auc.reshape(4,4)
cv_auc
```

Out[72]:

```
array([[0.63532748, 0.65089375, 0.70308491, 0.71966429],
       [0.67199386, 0.68045368, 0.71842984, 0.72627478],
       [0.67233895, 0.68212094, 0.71862575, 0.7255004 ],
       [0.66604735, 0.6786957 , 0.71492242, 0.72274055]])
```

In [73]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(cv_auc,annot=True)

plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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


plt.show()
```



In [74]:

```
# Testing the performance of the model on test data, plotting ROC Curves
# Select best log(C) value
best_set1_xgb = clfgbdt1.best_params_
print(best_set1_xgb)
```

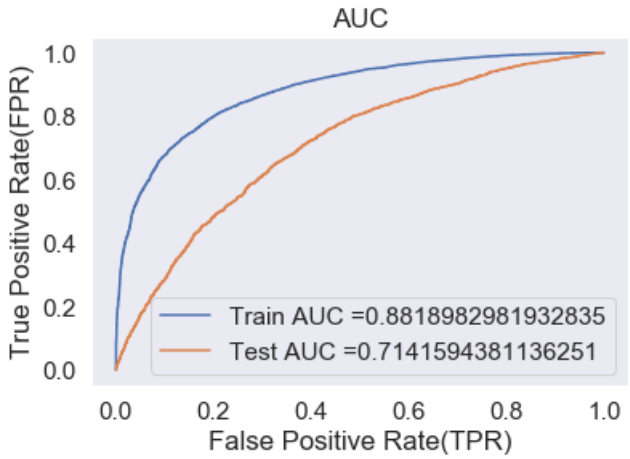
```
{'max_depth': 5, 'n_estimators': 100}
```

In [75]:

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

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

In [76]:

```
from sklearn.metrics import roc_curve, auc

gbdt1 = xgb.XGBClassifier(max_depth = 5, n_estimators = 100,n_jobs=-1,class_weight='balanced')
gbdt1.fit(X_train1, y_train)

y_train_pred = batch_predict(gbdt1, X_train1)
y_test_pred = batch_predict(gbdt1, X_test1)

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(FPR)")
plt.ylabel("True Positive Rate(TPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [77]:

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

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

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

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

In [78]:

```
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) 0.6399405130994342 for threshold 0.817
[[ 2873   738]
 [ 3882 15957]]
Test confusion matrix
```

```
the maximum value of tpr*(1-fpr) 0.4382784056925344 for threshold 0.845
[[1088  690]
 [2813 6959]]
```

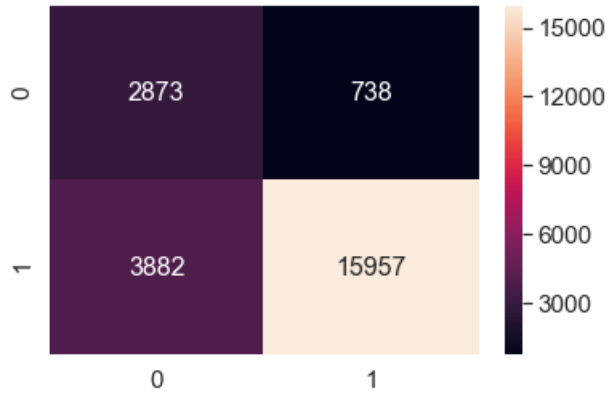
In [79]:

```
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) 0.6399405130994342 for threshold 0.817
```

Out[79]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f0b75cc0>
```



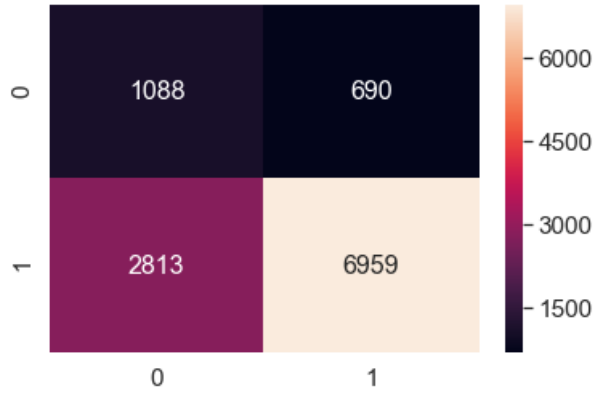
In [80]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4382784056925344 for threshold 0.845
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f0a02940>
```



2.5.2 Applying XGBOOST on TFIDF, SET 2

In [81]:

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

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
```

```
from scipy.sparse import hstack

X_train2=hstack((cat_0n_xtrain, cat_1p_xtrain, sub_cat_0n_xtrain, sub_cat_1p_xtrain,
school_state_0n_xtrain,
                 school_state_1p_xtrain, teacher_prefix_0n_xtrain, teacher_prefix_1p_xtrain,
                 proj_grade_0n_xtrain, proj_grade_1p_xtrain, price_standardized_xtrain,
                 teacher_num_prev_projects_standardized_xtrain,
                 quantity_standardized_xtrain,essay_tfidf_xtrain, proj_title_tfidf_xtrain)).tocsr()



X_test2=hstack((cat_0n_xtest, cat_1p_xtest, sub_cat_0n_xtest, sub_cat_1p_xtest,
school_state_0n_xtest,
                 school_state_1p_xtest, teacher_prefix_0n_xtest, teacher_prefix_1p_xtest,
                 proj_grade_0n_xtest, proj_grade_1p_xtest, price_standardized_xtest,
                teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                 essay_tfidf_xtest, proj_title_tfidf_xtest)).tocsr()


print(X_train2.shape)

print(X_test2.shape)
```

```
(23450, 6307)
(11550, 6307)
```

In [82]:

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


start_time = time.time()

gbdt2 = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [5, 10, 50, 100], 'max_depth':[2,  5, 7, 10]}
clfgbdt2 = GridSearchCV(gbdt2, parameters, cv= 3, scoring='roc_auc',return_train_score=True)
clfgbdt2.fit(X_train2, y_train)

train_auc= clfgbdt2.cv_results_['mean_train_score']
train_auc_std= clfgbdt2.cv_results_['std_train_score']
cv_auc = clfgbdt2.cv_results_['mean_test_score']
cv_auc_std= clfgbdt2.cv_results_['std_test_score']

print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 2275.0765404701233 ms
```

In [83]:

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

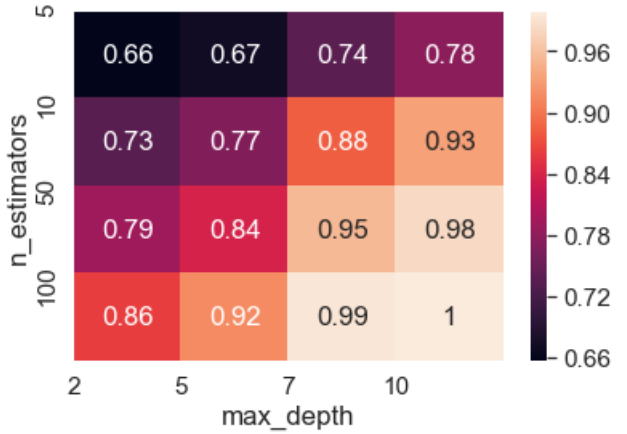
Out[83]:

```
array([0.6418029 , 0.6578974 , 0.70767283, 0.72280825, 0.67180781,
       0.68321147, 0.72371492, 0.72745293, 0.67145172, 0.68259737,
       0.72169252, 0.72592041, 0.66592586, 0.6786258 , 0.72068503,
       0.72551069])
```

In [84]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(4), [2,  5, 7, 10])
```

```
plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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


plt.show()
```



In [85]:

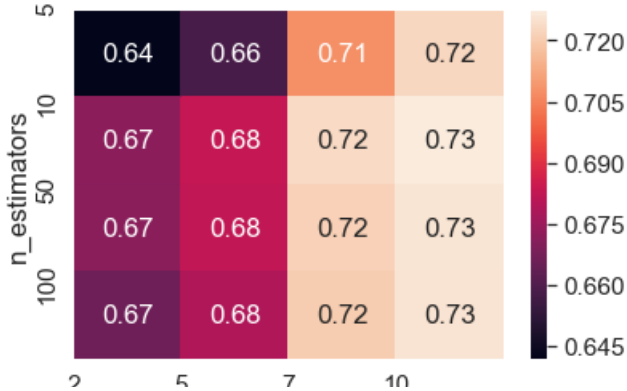
```
cv_auc = cv_auc.reshape(4,4)
cv_auc
```

Out[85]:

```
array([[0.6418029 , 0.6578974 , 0.70767283, 0.72280825],
       [0.67180781, 0.68321147, 0.72371492, 0.72745293],
       [0.67145172, 0.68259737, 0.72169252, 0.72592041],
       [0.66592586, 0.6786258 , 0.72068503, 0.72551069]])
```

In [86]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(cv_auc,annot=True)

plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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


plt.show()
```



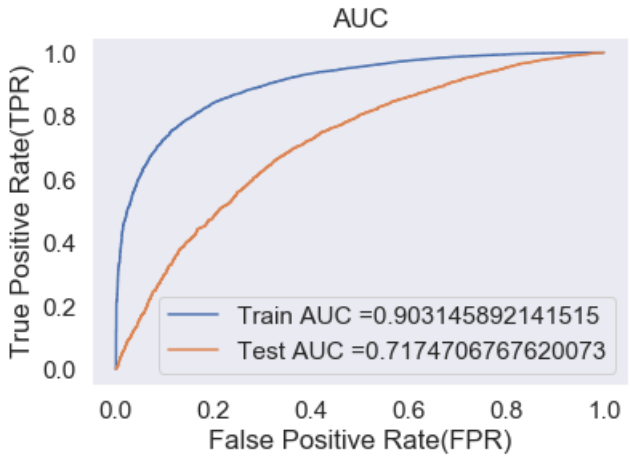
max_depth

In [101]:

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

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

In [87]:

```
from sklearn.metrics import roc_curve, auc

gbdt2 = xgb.XGBClassifier(max_depth = 5, n_estimators = 100,n_jobs=-1,class_weight='balanced')
gbdt2.fit(X_train2, y_train)

y_train_pred = batch_predict(gbdt2, X_train2)
y_test_pred = batch_predict(gbdt2, X_test2)

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(FPR)")
plt.ylabel("True Positive Rate(TPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [88]:

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

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

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

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

```
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [89]:

```
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) 0.6757059239645695 for threshold 0.821
[[ 3019   592]
 [ 3805 16034]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4456209564573974 for threshold 0.83
[[ 949  829]
 [2191 7581]]
```

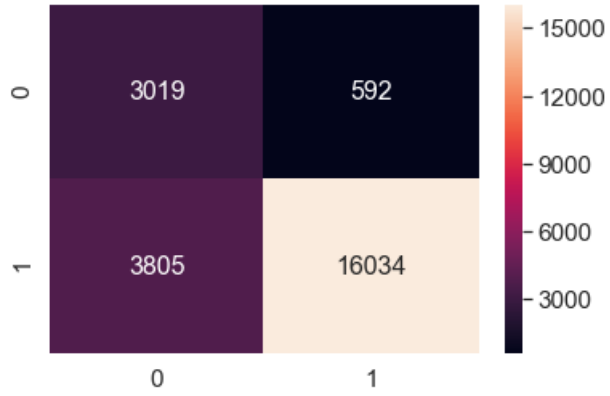
In [90]:

```
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) 0.6757059239645695 for threshold 0.821
```

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f1eda400>
```



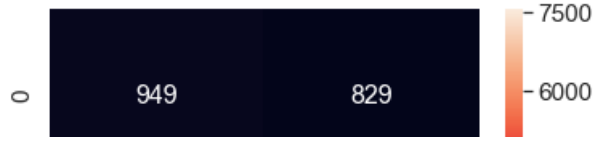
In [91]:

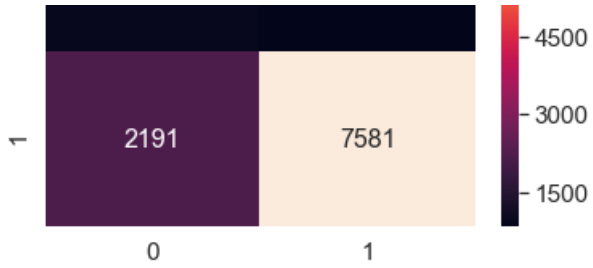
```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4456209564573974 for threshold 0.83
```

Out[91]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f1c05710>
```





2.5.3 Applying XGBOOST on AVG W2V, SET 3

In [92]:

```
essay_avg_w2v_vectors_xtrain= np.array(essay_avg_w2v_vectors_xtrain)
proj_title_avg_w2v_vectors_xtrain= np.array(proj_title_avg_w2v_vectors_xtrain)
```

In [93]:

```
essay_avg_w2v_vectors_xtest= np.array(essay_avg_w2v_vectors_xtest)
proj_title_avg_w2v_vectors_xtest= np.array(proj_title_avg_w2v_vectors_xtest)
```

In [94]:

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

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


X_train3=np.hstack((cat_0n_xtrain, cat_1p_xtrain, sub_cat_0n_xtrain, sub_cat_1p_xtrain,
school_state_0n_xtrain,
                  school_state_1p_xtrain, teacher_prefix_0n_xtrain, teacher_prefix_1p_xtrain,
                  proj_grade_0n_xtrain, proj_grade_1p_xtrain, price_standardized_xtrain,
                teacher_num_prev_projects_standardized_xtrain,
                 essay_avg_w2v_vectors_xtrain, proj_title_avg_w2v_vectors_xtrain))


X_test3=np.hstack((cat_0n_xtest, cat_1p_xtest, sub_cat_0n_xtest, sub_cat_1p_xtest,
school_state_0n_xtest,
                  school_state_1p_xtest, teacher_prefix_0n_xtest, teacher_prefix_1p_xtest,
                  proj_grade_0n_xtest, proj_grade_1p_xtest, price_standardized_xtest,
                teacher_num_prev_projects_standardized_xtest,
                 essay_avg_w2v_vectors_xtest, proj_title_avg_w2v_vectors_xtest))


print(X_train3.shape, y_train.shape)
print(X_test3.shape, y_test.shape)
```

```
(23450, 612) (23450,)
(11550, 612) (11550,)
```

GridsearchCV

In [95]:

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


start_time = time.time()

gbdt3 = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [5, 10, 50, 100], 'max_depth':[2,  5, 7, 10]}
clfgbdt3 = GridSearchCV(gbdt3, parameters, cv= 3, scoring='roc_auc',return_train_score=True)
clfgbdt3.fit(X_train3, y_train)

train_auc= clfgbdt3.cv_results_['mean_train_score']
```

```
train_auc= clfgbdt3.cv_results_['mean_train_score']
train_auc_std= clfgbdt3.cv_results_['std_train_score']
cv_auc = clfgbdt3.cv_results_['mean_test_score']
cv_auc_std= clfgbdt3.cv_results_['std_test_score']

print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 4714.778045892715 ms
```

In [96]:

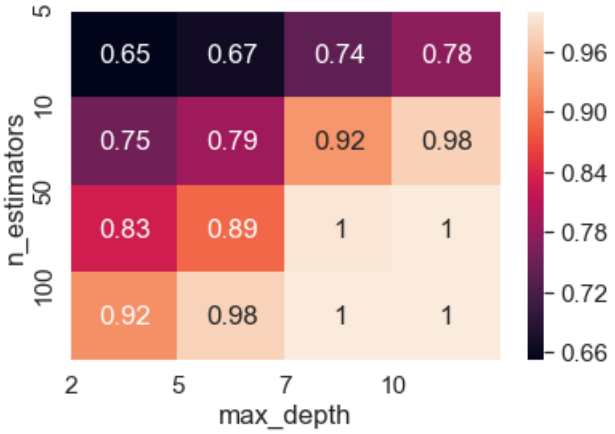
```
train_auc = train_auc.reshape(4,4)
train_auc
```

Out[96]:

```
array([[0.65254117, 0.66882663, 0.73687192, 0.77540914],
       [0.7520663 , 0.79047179, 0.92464503, 0.9762694 ],
       [0.83156365, 0.88984444, 0.99589256, 0.99994458],
       [0.91927932, 0.97740188, 1.        , 1.        ]])
```

In [97]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(train_auc,annot=True)

plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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


plt.show()
```



In [98]:

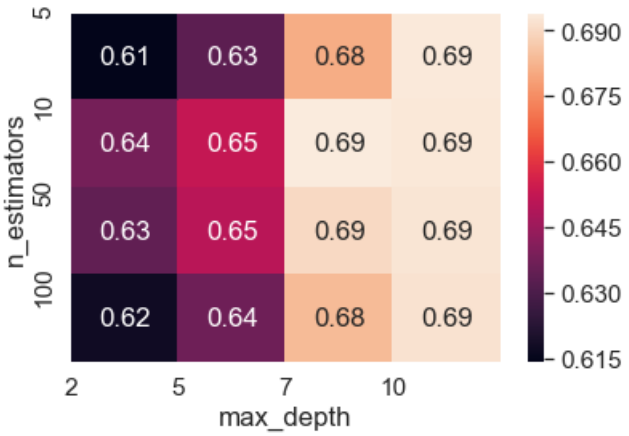
```
cv_auc = cv_auc.reshape(4,4)
cv_auc
```

Out[98]:

```
array([[0.61419759, 0.63344569, 0.68118149, 0.69361079],
       [0.63796571, 0.65260474, 0.69397146, 0.69322983],
       [0.63418111, 0.65036782, 0.69014853, 0.6924592 ],
       [0.61757766, 0.6369211 , 0.68387425, 0.69202393]])
```

In [99]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(cv_auc,annot=True)

plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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


plt.show()
```



In [102]:

```
# Testing the performance of the model on test data, plotting ROC Curves
# Select best log(C) value
best_set3_xgb = clfgbdt3.best_params_
print(best_set3_xgb)
```

```
{'max_depth': 5, 'n_estimators': 50}
```

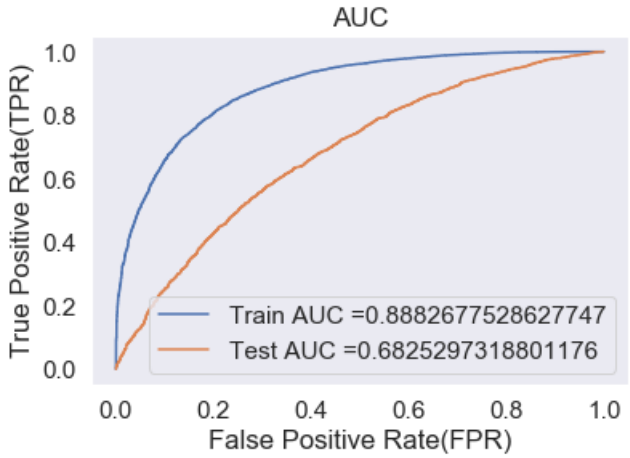
In [100]:

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

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

In [103]:

```
from sklearn.metrics import roc_curve, auc

gbdt3 = xgb.XGBClassifier(max_depth = 5, n_estimators = 50,n_jobs=-1,class_weight='balanced')
gbdt3.fit(X_train3, y_train)

y_train_pred = batch_predict(gbdt3, X_train3)
y_test_pred = batch_predict(gbdt3, X_test3)

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(FPR)")
plt.ylabel("True Positive Rate(TPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



In [106]:

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

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

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

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

In [107]:

```
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) 0.6472045968383902 for threshold 0.82
[[ 2870   741]
 [ 3684 16155]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4026774462238474 for threshold 0.84
[[1026  752]
 [3076 6696]]
```

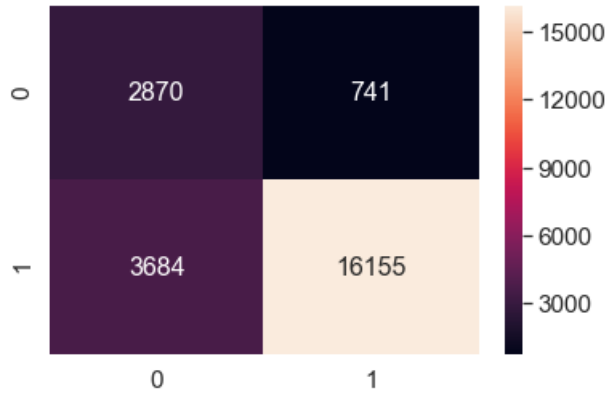
In [108]:

```
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) 0.6472045968383902 for threshold 0.82
```

Out[108]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c1daa9a860>
```



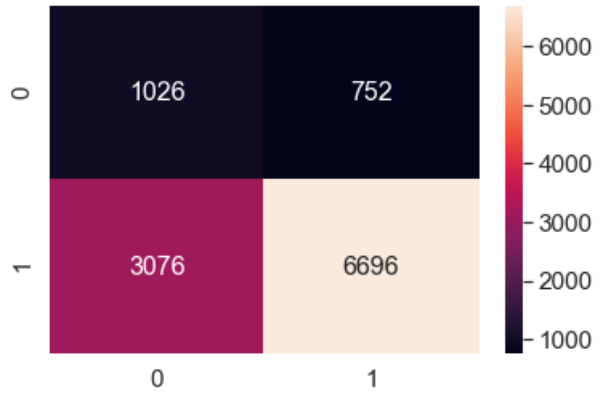
In [109]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4026774462238474 for threshold 0.84
```

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f3b31dd8>
```



2.5.4 Applying XGBOOST on TFIDF W2V, SET 4

In [110]:

```
essay_tfidf_w2v_vectors_xtrain=np.array(essay_tfidf_w2v_vectors_xtrain)
proj_title_tfidf_w2v_vectors_xtrain=np.array(proj_title_tfidf_w2v_vectors_xtrain)
```

In [111]:

```
essay_tfidf_w2v_vectors_xtest=np.array(essay_tfidf_w2v_vectors_xtest)
proj_title_tfidf_w2v_vectors_xtest=np.array(proj_title_tfidf_w2v_vectors_xtest)
```

In [112]:

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

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
```

```
from scipy.sparse import hstack


X_train4=np.hstack((cat_0n_xtrain, cat_1p_xtrain, sub_cat_0n_xtrain, sub_cat_1p_xtrain,
school_state_0n_xtrain,
                 school_state_1p_xtrain, teacher_prefix_0n_xtrain, teacher_prefix_1p_xtrain,
                 proj_grade_0n_xtrain, proj_grade_1p_xtrain, price_standardized_xtrain,
                teacher_num_prev_projects_standardized_xtrain, quantity_standardized_xtrain,
                 essay_tfidf_w2v_vectors_xtrain, proj_title_tfidf_w2v_vectors_xtrain))



X_test4=np.hstack((cat_0n_xtest, cat_1p_xtest, sub_cat_0n_xtest, sub_cat_1p_xtest,
school_state_0n_xtest,
                 school_state_1p_xtest, teacher_prefix_0n_xtest, teacher_prefix_1p_xtest,
                 proj_grade_0n_xtest, proj_grade_1p_xtest, price_standardized_xtest,
                teacher_num_prev_projects_standardized_xtest, quantity_standardized_xtest,
                 essay_tfidf_w2v_vectors_xtest, proj_title_tfidf_w2v_vectors_xtest))


print(X_train4.shape, y_train.shape)

print(X_test4.shape, y_test.shape)
```

```
(23450, 613) (23450,)
(11550, 613) (11550,)
```

GridsearchCV

In [113]:

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


start_time = time.time()

gbdt4 = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [5, 10, 50, 100], 'max_depth':[2,  5, 7, 10]}
clfgbdt4 = GridSearchCV(gbdt4, parameters, cv= 3, scoring='roc_auc',return_train_score=True)
clfgbdt4.fit(X_train4, y_train)

train_auc= clfgbdt4.cv_results_['mean_train_score']
train_auc_std= clfgbdt4.cv_results_['std_train_score']
cv_auc = clfgbdt4.cv_results_['mean_test_score']
cv_auc_std= clfgbdt4.cv_results_['std_test_score']

print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

```
Execution time: 4091.337624311447 ms
```

In [114]:

```
import dill
dill.dump_session('notebook_71_11.db')
#dill.load_session('notebook_71_11.db')
```

In [116]:

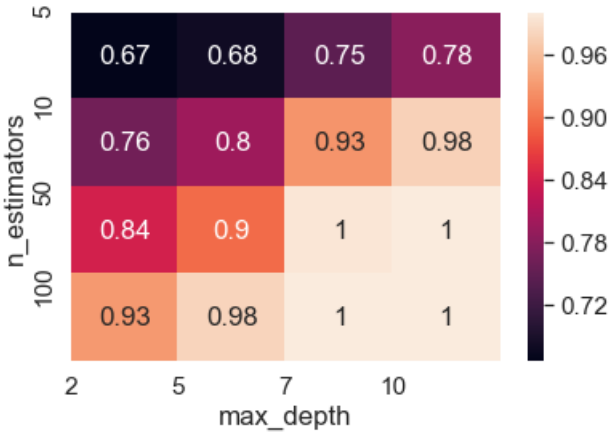
```
train_auc = train_auc.reshape(4,4)
train_auc
```

Out[116]:

```
array([[0.66661044, 0.68069013, 0.74620316, 0.78144917],
       [0.76396476, 0.79750758, 0.92654574, 0.97697661],
       [0.84006529, 0.89611936, 0.99579526, 0.99997085],
       [0.93076875, 0.9802885 , 1.        , 1.        ]])
```

In [117]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(train_auc,annot=True)

plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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


plt.show()
```



In [118]:

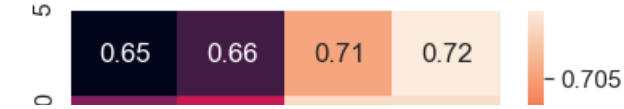
```
cv_auc = cv_auc.reshape(4,4)
cv_auc
```

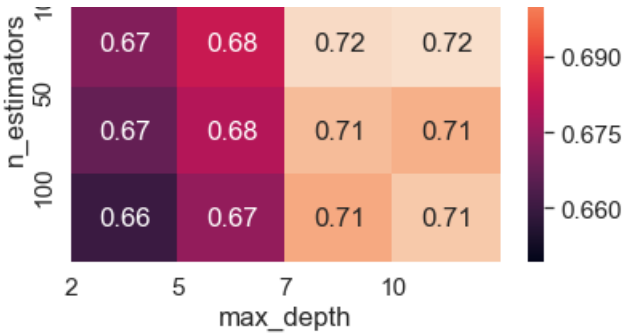
Out[118]:

```
array([[0.64930629, 0.66130362, 0.70617867, 0.71859382],
       [0.67209494, 0.68349204, 0.71537926, 0.71619433],
       [0.66736195, 0.67996791, 0.71015428, 0.7080019 ],
       [0.66012483, 0.67487744, 0.70686212, 0.71231797]])
```

In [119]:

```
import matplotlib.pyplot as plt

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


sns.heatmap(cv_auc,annot=True)

plt.xticks(np.arange(4), [2,  5, 7, 10])
plt.yticks(np.arange(4), [5, 10, 50, 100])


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


plt.show()
```





In [120]:

```
# Testing the performance of the model on test data, plotting ROC Curves
# Select best log(C) value
best_set4_xgb = clfgbdt4.best_params_
print(best_set4_xgb)
```

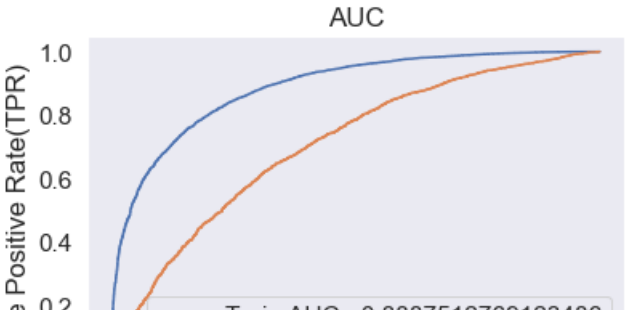
```
{'max_depth': 2, 'n_estimators': 100}
```

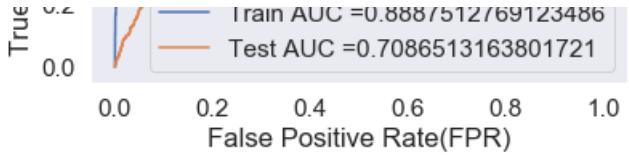
In [121]:

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

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

In [122]:

```
from sklearn.metrics import roc_curve, auc

gbdt4 = xgb.XGBClassifier(max_depth = 5, n_estimators = 50,n_jobs=-1,class_weight='balanced')
gbdt4.fit(X_train4, y_train)

y_train_pred = batch_predict(gbdt4, X_train4)
y_test_pred = batch_predict(gbdt4, X_test4)

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(FPR)")
plt.ylabel("True Positive Rate(TPR)")
plt.title("AUC")
plt.grid()
plt.show()
```





In [123]:

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

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

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

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

In [124]:

```
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) 0.6438508475643776 for threshold 0.82
[[ 2896   715]
 [ 3912 15927]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.42982619011551104 for threshold 0.841
[[1106  672]
 [3148 6624]]
```

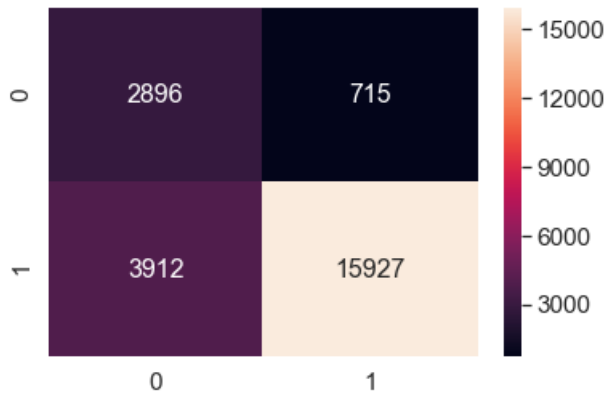
In [125]:

```
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) 0.6438508475643776 for threshold 0.82
```

Out[125]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f1b556d8>
```



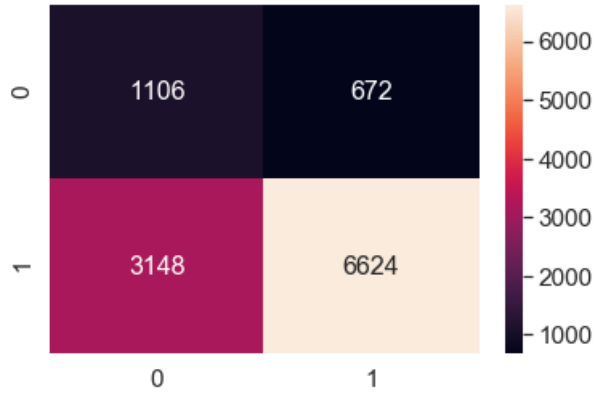
In [126]:

```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, t
est_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.42982619011551104 for threshold 0.841
```

Out[126]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1c0f19ae940>
```



3. Conclusion

In [ ]:

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

In [127]:

```
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]", 0.70001])
x.add_row(["TFIDF", "RF", "[10,100]", 0.6903])
x.add_row(["AVG W2V", "RF", "[10,10]", 0.61638])
x.add_row(["TFIDF W2V", "RF", "[10,100]", 0.69172])
x.add_row(["BOW", "GBDT","[5,100]", 0.71415])
x.add_row(["TFIDF", "GBDT", "[5,100]", 0.71747])
x.add_row(["AVG W2V", "GBDT", "[5,50]", 0.6825])
x.add_row(["TFIDF W2V", "GBDT", "[5,50]", 0.70865])

print(x)
```

```
+------------+-------+-----------------------------------------+----------+
| Vectorizer | Model | Hyperparameters[max_depth,n_estimators] | Test AUC |
+------------+-------+-----------------------------------------+----------+
|    BOW     |   RF  |                 [10,500]                | 0.70001  |
|   TFIDF    |   RF  |                 [10,100]                |  0.6903  |
|  AVG W2V   |   RF  |                 [10,10]                 | 0.61638  |
| TFIDF W2V  |   RF  |                 [10,100]                | 0.69172  |
|    BOW     |  GBDT |                 [5,100]                 | 0.71415  |
|   TFIDF    |  GBDT |                 [5,100]                 | 0.71747  |
|  AVG W2V   |  GBDT |                  [5,50]                 |  0.6825  |
| TFIDF W2V  |  GBDT |                  [5,50]                 | 0.70865  |
+------------+-------+-----------------------------------------+----------+
```

In [ ]:

```
```