RFGBDT and XGBOOST

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://acco
unts.google.com/o/oauth2/auth?client_id=9
47318989803-6bn6qk8qdgf4n4g3pfee6491hc0br
c4i.apps.googleusercontent.com&redirect_u
ri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&re
sponse_type=code&scope=email%20https%3a%2
f%2fwww.googleapis.com%2fauth%2fdocs.tes
t%20https%3a%2f%2fwww.googleapis.com%2fau
th%2fdrive%20https%3a%2f%2fwww.googleapi
s.com%2fauth%2fdrive.photos.readonly%20ht
tps%3a%2f%2fwww.googleapis.com%2fauth%2fp
eopleapi.readonly

Enter your authorization code:
··········
Mounted at /content/drive
```

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	De
`project_id`	A u pro p03
`project_title`	Titl • •

Feature	De
project_grade_category	Gr the foll • • • •
project_subject_categories	On sul fro of \\ • • • • • • • • • **Ex** • •
school_state	Sta (Tv **Ex**

Feature	De
`project_subject_subcategories`	On sub pro • •
`project_resource_summary`	An nee •
`project_essay_1`	Fir
`project_essay_2`	Se
`project_essay_3`	Thi
`project_essay_4`	Fo
`project_submitted_datetime`	Da wa `04-`
`teacher_id`	A u of t `bdf`

Feature	De
`teacher_prefix`	Tea foll • • • • • •
`teacher_number_of_previously_posted_projects`	Nu pre tea

* 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 [2]:
```
!pip install chart_studio
```

```
Requirement already satisfied: chart_stud
io in /usr/local/lib/python3.6/dist-packa
ges (1.0.0)
Requirement already satisfied: requests i
n /usr/local/lib/python3.6/dist-packages
(from chart_studio) (2.21.0)
Requirement already satisfied: six in /us
r/local/lib/python3.6/dist-packages (from
chart_studio) (1.12.0)
Requirement already satisfied: plotly in
/usr/local/lib/python3.6/dist-packages (f
rom chart_studio) (4.1.1)
Requirement already satisfied: retrying>=
1.3.3 in /usr/local/lib/python3.6/dist-pa
ckages (from chart_studio) (1.3.3)
Requirement already satisfied: chardet<3.
1.0,>=3.0.2 in /usr/local/lib/python3.6/d
ist-packages (from requests->chart_studi
o) (3.0.4)
Requirement already satisfied: certifi>=2
017.4.17 in /usr/local/lib/python3.6/dist
-packages (from requests->chart_studio)
(2019.11.28)
Requirement already satisfied: idna<2.9,>
```

```
=2.5 in /usr/local/lib/python3.6/dist-pac
kages (from requests->chart_studio) (2.8)
Requirement already satisfied: urllib3<1.
25,>=1.21.1 in /usr/local/lib/python3.6/d
ist-packages (from requests->chart_studi
o) (1.24.3)
```

In [3]:

```
%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 Tfi
dfTransformer
from sklearn.feature_extraction.text import Tfi
dfVectorizer

from sklearn.feature_extraction.text import Cou
ntVectorizer
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: ht
tps://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



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

1.1 Reading Data

In [0]:

```
project_data = pd.read_csv('/content/drive/My Drive/Assignments_DonorsChoose_2018/train_data.csv',nrows=30000)
resource_data = pd.read_csv('/content/drive/My Drive/Assignments_DonorsChoose_2018/resources.csv')
```

In [7]:

```
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 (30000, 17)
```

```
-----------------------------------------------------------
---------
The attributes of data : ['Unnamed: 0' 'i
d' 'teacher_id' 'teacher_prefix' 'school_
state'
 'project_submitted_datetime' 'project_gr
ade_category'
 'project_subject_categories' 'project_su
bject_subcategories'
 'project_title' 'project_essay_1' 'proje
ct_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summ
ary'
 'teacher_number_of_previously_posted_pro
jects' 'project_is_approved']
```

In [8]:

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

```
Number of data points in train data (1541
272, 4)
['id' 'description' 'quantity' 'price']
```

Out[8]:

	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

1.2 preprocessing of `project_subject_categories`

In [0]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & S
```

```
cience"=> "Math","&", "Science"
            j=j.replace('The','') # if we have
 the words "The" we are going to replace it wit
h ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing
all the ' '(space) with ''(empty) ex:"Math & Sc
ience"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() wi
ll return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are r
eplacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [0]:

```
sub_catogories = list(project_data['project_sub
ject_subcategories'].values)
# remove special characters from list of string
```

```python
s 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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

In [0]:

```
# We need to get rid of The spaces between the
 text and the hyphens because they're special c
haracters.
#Rmoving multiple characters from a string in P
ython
#https://stackoverflow.com/questions/3411771/mu
ltiple-character-replace-with-python

project_grade_category = []

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

In [12]:

```
project_data.drop(['project_grade_category'], a
xis = 1, inplace = True)
project_data["project_grade_category"] = projec
t_grade_category
print("After removing the special characters ,C
olumn values:  ")
np.unique(project_data["project_grade_category"
].values)
```

```
After removing the special characters ,Co
lumn values:
```

Out[12]:
```
array(['Grades_3_5', 'Grades_6_8', 'Grade
s_9_12', 'Grades_PreK_2'],
      dtype=object)
```

In [0]:
```
#NaN values in techer prefix will create a prob
lem while encoding,so we replace NaN values wit
h the mode of that particular column
#removing dot(.) since it is a special characte
r
mode_of_teacher_prefix = project_data['teacher_
prefix'].value_counts().index[0]


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

In [0]:
```
prefixes = []

for i in range(len(project_data)):
    a = project_data["teacher_prefix"][i].repla
ce(".", "")
    prefixes.append(a)
```

In [15]:
```
project_data.drop(['teacher_prefix'], axis = 1,
inplace = True)
project_data["teacher_prefix"] = prefixes
print("After removing the special characters ,C
olumn values:  ")
np.unique(project_data["teacher_prefix"].values
)
```

```
After removing the special characters ,Co
```

lumn values:

Out[15]:

```
array(['Mr', 'Mrs', 'Ms', 'Teacher'], dty
pe=object)
```

1.3 Text preprocessing

In [0]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_e
ssay_1"].map(str) +\
                        project_data["project_e
ssay_2"].map(str) + \
                        project_data["project_e
ssay_3"].map(str) + \
                        project_data["project_e
ssay_4"].map(str)
```

In [0]:

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

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

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

```
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [18]:

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

```
My kindergarten students have varied disa
bilities ranging from speech and language
delays, cognitive delays, gross/fine moto
r delays, to autism. They are eager beave
rs and always strive to work their hardes
t working past their limitations. \r\n\r
\nThe materials we have are the ones I se
ek out for my students. I teach in a Titl
e I school where most of the students rec
eive free or reduced price lunch.  Despit
e their disabilities and limitations, my
students love coming to school and come e
ager to learn and explore.Have you ever f
elt like you had ants in your pants and y
ou needed to groove and move as you were
in a meeting? This is how my kids feel al
l the time. The want to be able to move a
s they learn or so they say.Wobble chairs
are the answer and I love then because th
ey develop their core, which enhances gro
ss motor and in Turn fine motor skills.
\r\nThey also want to learn through game
s, my kids do not want to sit and do work
sheets. They want to learn to count by ju
mping and playing. Physical engagement is
the key to our success. The number toss a
nd color and shape mats can make that hap
```

```
pen. My students will forget they are doi
ng work and just have the fun a 6 year ol
d deserves.nannan
==========================================
=========
```

In [19]:

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

```
My kindergarten students have varied disa
bilities ranging from speech and language
delays, cognitive delays, gross/fine moto
r delays, to autism. They are eager beave
rs and always strive to work their hardes
t working past their limitations.     The
materials we have are the ones I seek out
for my students. I teach in a Title I sch
ool where most of the students receive fr
ee or reduced price lunch.  Despite their
disabilities and limitations, my students
love coming to school and come eager to l
earn and explore.Have you ever felt like
you had ants in your pants and you needed
to groove and move as you were in a meeti
ng? This is how my kids feel all the tim
e. The want to be able to move as they le
arn or so they say.Wobble chairs are the
answer and I love then because they devel
op their core, which enhances gross motor
and in Turn fine motor skills.    They als
o want to learn through games, my kids do
not want to sit and do worksheets. They w
```

```
ant to learn to count by jumping and play
ing. Physical engagement is the key to ou
r success. The number toss and color and 
shape mats can make that happen. My stude
nts will forget they are doing work and j
ust have the fun a 6 year old deserves.na
nnan
```

In [20]:

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

```
My kindergarten students have varied disa
bilities ranging from speech and language
delays cognitive delays gross fine motor 
delays to autism They are eager beavers a
nd always strive to work their hardest wo
rking past their limitations The material
s we have are the ones I seek out for my 
students I teach in a Title I school wher
e most of the students receive free or re
duced price lunch Despite their disabilit
ies and limitations my students love comi
ng to school and come eager to learn and 
explore Have you ever felt like you had a
nts in your pants and you needed to groov
e and move as you were in a meeting This 
is how my kids feel all the time The want
to be able to move as they learn or so th
ey say Wobble chairs are the answer and I
love then because they develop their core
which enhances gross motor and in Turn fi
ne motor skills They also want to learn t
hrough games my kids do not want to sit a
nd do worksheets They want to learn to co
```

```
unt by jumping and playing Physical engag
ement is the key to our success The numbe
r toss and color and shape mats can make
that happen My students will forget they
are doing work and just have the fun a 6
year old deserves nannan
```

In [0]:

```
# 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', 'w
```

```
hen', 'where', 'why', 'how', 'all', 'any', 'bot
h', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'o
nly', 'own', 'same', 'so', 'than', 'too', 'ver
y', \
            's', 't', 'can', 'will', 'just', 'd
on', "don't", 'should', "should've", 'now', 'd'
, 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't",
'couldn', "couldn't", 'didn', "didn't", 'doesn'
, "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven'
, "haven't", 'isn', "isn't", 'ma', 'mightn', "m
ightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'sha
n', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "would
n't"]
```

In [0]:

```
#convert all the words to lower case first and
 then remove the stopwords
for i in range(len(project_data['essay'].values
)):
    project_data['essay'].values[i] = project_d
ata['essay'].values[i].lower()
```

In [23]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].valu
es):
    sent = decontracted(sentance)
```

```
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = sent.replace('nan',' ')
    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%|██████████| 30000/30000 [00:13<00:00, 2235.93it/s]
```

In [0]:

```
#creating a new column with the preprocessed essays and replacing it with the original columns
project_data['preprocessed_essays'] = preprocessed_essays
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
```

In [0]:

```
essay_word_count=[]
for i in range(len(project_data['preprocessed_essays'])):
    essay_word_count.append(len(project_data['preprocessed_essays'][i].split()))
```

In [0]:

```
project_data['essay_word_count'] = essay_word_c
```

```
ount
```

1.4 Preprocessing of $project_tit$

In [0]:
```
#convert all the words to lower case first and then remove the stopwords
for i in range(len(project_data['project_title'].values)):
    project_data['project_title'].values[i] = project_data['project_title'].values[i].lower()
```

In [28]:
```
# similarly you can preprocess the titles also
preprocessed_titles = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['project_title'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = sent.replace('nan',' ')
    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_titles.append(sent.lower().strip())
```

```
100%|██████████| 30000/30000 [00:00<00:00, 51017.07it/s]
```

In [0]:
```
#creating a new column with the preprocessed ti
```

```
tles,useful for analysis
project_data['preprocessed_titles'] = preproces
sed_titles
```

In [0]:

```
title_word_count=[]
for i in range(len(project_data['preprocessed_t
itles'])):
    title_word_count.append(len(project_data['p
reprocessed_titles'][i].split()))
```

In [0]:

```
project_data['title_word_count'] = title_word_c
ount
```

In [32]:

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

```
[nltk_data] Downloading package vader_lex
icon to /root/nltk_data...
```

Out[32]:

```
True
```

In [33]:

```
import nltk
from nltk.sentiment.vader import SentimentInten
sityAnalyzer
analyzer = SentimentIntensityAnalyzer()
neg=[];pos=[];neu=[]; compound = []

for i in tqdm(range(len(project_data['preproces
sed_essays']))):
    sentiment_scores = analyzer.polarity_scores
(project_data['preprocessed_essays'][i])
    neg.append(sentiment_scores['neg'])
    pos.append(sentiment_scores['pos'])
    neu.append(sentiment_scores['neu'])
```

```
    compound.append(sentiment_scores['compound'])
```

```
100%|██████████| 30000/30000 [00:44<00:00, 678.41it/s]
```

In [0]:

```
#new columns indicating the sentiment score of each project essay
project_data['neg'] = neg
project_data['neu'] = neu
project_data['pos'] = pos
project_data['compound'] = compound
```

Splitting data into Train and test: Stratified Sampling

In [0]:

```
# train test split

from sklearn.model_selection import train_test_split

project_data_train, project_data_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'])
```

In [36]:

```
print("Split ratio")
print('-'*50)
print('Train dataset:',len(project_data_train)/len(project_data)*100,'%\n','size:',len(project_data_train))
print('Test dataset:',len(project_data_test)/le
```

```
n(project_data)*100,'%\n','size:',len(project_d
ata_test))
```

```
Split ratio
-----------------------------------------
---------
Train dataset: 67.0 %
 size: 20100
Test dataset: 33.0 %
 size: 9900
```

In [0]:

```
#Features
project_data_train.drop(['project_is_approved'
], axis=1, inplace=True)

project_data_test.drop(['project_is_approved'],
axis=1, inplace=True)
```

1.5 Preparing data for models

In [38]:

```
project_data.columns
```

Out[38]:

```
Index(['Unnamed: 0', 'id', 'teacher_id',
'school_state',
       'project_submitted_datetime', 'pro
ject_title',
       'project_resource_summary',
       'teacher_number_of_previously_post
ed_projects', 'project_is_approved',
       'clean_categories', 'clean_subcate
gories', 'project_grade_category',
       'teacher_prefix', 'essay', 'prepro
cessed_essays', 'essay_word_count',
       'preprocessed_titles', 'title_word
```

```
_count', 'neg', 'neu', 'pos',
       'compound'],
      dtype='object')
```

we are going to consider

```
   - school_state : categorical data
   - clean_categories : categorical data
   - clean_subcategories : categorical d
ata
   - project_grade_category : categorica
l data
   - teacher_prefix : categorical data

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

   - quantity : numerical (optinal)
   - teacher_number_of_previously_posted
_projects : numerical
   - price : numerical
```

Make Data Model Ready: vectorizing numerical, categorical features (with response coding)

Make Data Model Ready: encoding eassay, and project_title

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [39]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(project_data_train['preprocessed_essays'].values)  #Fitting has to be on Train data


train_essay_bow = vectorizer_bow_essay.transform(project_data_train['essay'].values)

test_essay_bow = vectorizer_bow_essay.transform(project_data_test['essay'].values)


print("Shape of train data matrix after one hot encoding ",train_essay_bow.shape)

print("Shape of test data matrix after one hot encoding ",test_essay_bow.shape)
```

```
Shape of train data matrix after one hot

encoding  (20100, 8434)
Shape of test data matrix after one hot encoding  (9900, 8434)
```

In [40]:

```
# you can vectorize the title also
# before you vectorize the title make sure you
```

```
 preprocess it
vectorizer_bow_title = CountVectorizer(min_df=1
0)
vectorizer_bow_title.fit_transform(project_data
_train['preprocessed_titles'].values)     #Fitti
ng has to be on Train data



train_title_bow = vectorizer_bow_title.transfor
m(project_data_train['preprocessed_titles'].val
ues)

test_title_bow = vectorizer_bow_title.transform
(project_data_test['preprocessed_titles'].value
s)




print("Shape of train data matrix after one hot
encoding ",train_title_bow.shape)

print("Shape of test data matrix after one hot
 encoding ",test_title_bow.shape)
```

```
Shape of train data matrix after one hot
encoding  (20100, 1063)
Shape of test data matrix after one hot e
ncoding  (9900, 1063)
```

1.5.2.2 TFIDF vectorizer

In [41]:

```
from sklearn.feature_extraction.text import Tfi
dfVectorizer
vectorizer_tfidf_essay = TfidfVectorizer(min_df
```

```
=10)
vectorizer_tfidf_essay.fit(project_data_train[
'preprocessed_essays'])      #Fitting has to be
on Train data

train_essay_tfidf = vectorizer_tfidf_essay.tran
sform(project_data_train['preprocessed_essays']
.values)

test_essay_tfidf = vectorizer_tfidf_essay.trans
form(project_data_test['preprocessed_essays'].v
alues)

print("Shape of train data matrix after one hot
encoding ",train_essay_tfidf.shape)

print("Shape of test data matrix after one hot
 encoding ",test_essay_tfidf.shape)
```

```
Shape of train data matrix after one hot
encoding  (20100, 8434)
Shape of test data matrix after one hot e
ncoding  (9900, 8434)
```

In [42]:

```
vectorizer_tfidf_title = TfidfVectorizer(min_df
=10)
vectorizer_tfidf_title.fit(project_data_train[
'preprocessed_titles'])      #Fitting has to be
on Train data

train_title_tfidf = vectorizer_tfidf_title.tran
sform(project_data_train['preprocessed_titles']
.values)

test_title_tfidf = vectorizer_tfidf_title.trans
form(project_data_test['preprocessed_titles'].v
```

```
alues)

print("Shape of train data matrix after one hot
encoding ",train_title_tfidf.shape)

print("Shape of test data matrix after one hot
 encoding ",test_title_tfidf.shape)
```

```
Shape of train data matrix after one hot
encoding  (20100, 1063)
Shape of test data matrix after one hot e
ncoding  (9900, 1063)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [43]:

```
'''
# Reading glove vectors in python: https://stac
koverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for va
l in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

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

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

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

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

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

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


# stronging variables into pickle files python:
http://www.jessicayung.com/how-to-use-pickle-to
```

```
-save-and-load-variables-in-python/

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


'''
```

Out[43]:

```
'\n# Reading glove vectors in python: htt
ps://stackoverflow.com/a/38230349/4084039
\ndef loadGloveModel(gloveFile):\n    pri
nt ("Loading Glove Model")\n    f = open
(gloveFile,\'r\', encoding="utf8")\n    m
odel = {}\n    for line in tqdm(f):\n
splitLine = line.split()\n        word =
splitLine[0]\n        embedding = np.arra
y([float(val) for val in splitLine[1:]])
\n        model[word] = embedding\n    pr
int ("Done.",len(model)," words loaded!")
\n    return model\nmodel = loadGloveMode
l(\'glove.42B.300d.txt\')\n\n# ==========
==================\nOutput:\n    \nLoadin
g Glove Model\n1917495it [06:32, 4879.69i
t/s]\nDone. 1917495  words loaded!\n\n# =
===========================\n\nwords = []
\nfor i in preproced_texts:\n    words.ex
tend(i.split(\' \'))\n\nfor i in preproce
d_titles:\n    words.extend(i.split(\'
\'))\nprint("all the words in the coupu
s", len(words))\nwords = set(words)\nprin
t("the unique words in the coupus", len(w
ords))\n\ninter_words = set(model.keys
()).intersection(words)\nprint("The numbe
r of words that are present in both glove
vectors and our coupus",       len(inter_
```

```
words),"(",np.round(len(inter_words)/len
(words)*100,3),"%)")\n\nwords_courpus =
{}\nwords_glove = set(model.keys())\nfor
i in words:\n    if i in words_glove:\n
words_courpus[i] = model[i]\nprint("word
2 vec length", len(words_courpus))\n\n\n#
stronging variables into pickle files pyt
hon: http://www.jessicayung.com/how-to-us
e-pickle-to-save-and-load-variables-in-py
thon/\n\nimport pickle\nwith open(\'glove
_vectors\', \'wb\') as f:\n    pickle.dum
p(words_courpus, f)\n\n\n'
```

In [0]:

```python
# 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('/content/drive/My Drive/Assignments_DonorsChoose_2018/glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [45]:

```python
# average Word2Vec
# compute average word2vec for each review.
train_avg_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_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
    train_avg_w2v_essays.append(vector)

print(len(train_avg_w2v_essays))
print(len(train_avg_w2v_essays[0]))
```

```
100%|██████████| 20100/20100 [00:04<00:00, 4623.84it/s]
```

```
20100
300
```

In [46]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_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
    test_avg_w2v_essays.append(vector)
```

```
print(len(test_avg_w2v_essays))
print(len(test_avg_w2v_essays[0]))
```

```
100%|██████████| 9900/9900 [00:02<00:00, 4757.38it/s]
```

```
9900
300
```

In [47]:

```
# average Word2Vec
# compute average word2vec for each review.
train_avg_w2v_titles = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_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
    train_avg_w2v_titles.append(vector)

print(len(train_avg_w2v_titles))
print(len(train_avg_w2v_titles[0]))
```

```
100%|██████████| 20100/20100 [00:00<00:00, 81217.62it/s]
```

```
20100
300
```

In [48]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_titles = []; # the avg-w2v for eac
h sentence/review is stored in this list
for sentence in tqdm(project_data_test['preproc
essed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors ar
e of zero length
    cnt_words =0; # num of words with a valid v
ector in the sentence/review
    for word in sentence.split(): # for each wo
rd in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    test_avg_w2v_titles.append(vector)

print(len(test_avg_w2v_titles))
print(len(test_avg_w2v_titles[0]))
```

```
100%|██████████| 9900/9900 [00:00<00:00,
81002.08it/s]
```

```
9900
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [0]:

```
# S = ["abc def pqr", "def def def abc", "pqr p
```

```
qr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocesse
d_essays'].values)
# we are converting a dictionary with word as a
key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_n
ames(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names
())
```

In [50]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_essays = []; # the avg-w2v for
 each sentence/review is stored in this list
for sentence in tqdm(project_data_train['prepro
cessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors ar
e of zero length
    tf_idf_weight =0; # num of words with a val
id vector in the sentence/review
    for word in sentence.split(): # for each wo
rd in a review/sentence
        if (word in glove_words) and (word in t
fidf_words):
            vec = model[word] # getting the vec
tor for each word
            # here we are multiplying idf value
(dictionary[word]) and the tf value((sentence.c
ount(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence
.count(word)/len(sentence.split())) # getting t
he tfidf value for each word
            vector += (vec * tf_idf) # calculat
ing tfidf weighted w2v
```

```
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_essays.append(vector)

print(len(train_tfidf_w2v_essays))
print(len(train_tfidf_w2v_essays[0]))
```

```
100%|██████████| 20100/20100 [00:28<00:0
0, 711.79it/s]
```

```
20100
300
```

In [51]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_essays = []; # the avg-w2v for e
ach sentence/review is stored in this list
for sentence in tqdm(project_data_test['preproc
essed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors ar
e of zero length
    tf_idf_weight =0; # num of words with a val
id vector in the sentence/review
    for word in sentence.split(): # for each wo
rd in a review/sentence
        if (word in glove_words) and (word in t
fidf_words):
            vec = model[word] # getting the vec
tor for each word
            # here we are multiplying idf value
(dictionary[word]) and the tf value((sentence.c
ount(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence
.count(word)/len(sentence.split())) # getting t
```

```
he tfidf value for each word
            vector += (vec * tf_idf) # calculat
ing tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_essays.append(vector)

print(len(test_tfidf_w2v_essays))
print(len(test_tfidf_w2v_essays[0]))
```

```
100%|██████████| 9900/9900 [00:13<00:00,
714.70it/s]
```

```
9900
300
```

In [0]:

```
# Similarly you can vectorize for title also
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocesse
d_titles'])
# we are converting a dictionary with word as a
key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_n
ames(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names
())
```

In [53]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_titles = []; # the avg-w2v for
 each sentence/review is stored in this list
for sentence in tqdm(project_data_train['prepro
cessed_titles']): # for each review/sentence
    vector = np.zeros(300) # as word vectors ar
```

```
e of zero length
    tf_idf_weight =0; # num of words with a val
id vector in the sentence/review
    for word in sentence.split(): # for each wo
rd in a review/sentence
        if (word in glove_words) and (word in t
fidf_words):
            vec = model[word] # getting the vec
tor for each word
            # here we are multiplying idf value
(dictionary[word]) and the tf value((sentence.c
ount(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence
.count(word)/len(sentence.split())) # getting t
he tfidf value for each word
            vector += (vec * tf_idf) # calculat
ing tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_titles.append(vector)

print(len(train_tfidf_w2v_titles))
print(len(train_tfidf_w2v_titles[0]))
```

```
100%|██████████| 20100/20100 [00:00<00:0
0, 46103.60it/s]
```

```
20100
300
```

In [54]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_titles = []; # the avg-w2v for e
ach sentence/review is stored in this list
```

```
for sentence in tqdm(project_data_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 tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_titles.append(vector)

print(len(test_tfidf_w2v_titles))
print(len(test_tfidf_w2v_titles[0]))
```

```
100%|██████████| 9900/9900 [00:00<00:00, 42286.93it/s]
```

```
9900
300
```

1.5.3 Vectorizing Numerical features

In [0]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
```

In [0]:

```
project_data_train = pd.merge(project_data_train, price_data, on='id', how='left')

project_data_test = pd.merge(project_data_test, price_data, on='id', how='left')
```

In [57]:

```
from sklearn.preprocessing import Normalizer
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.

normalizer = Normalizer()
normalizer.fit(project_data_train['price'].values.reshape(1,-1))



price_normalized_train = normalizer.transform(project_data_train['price'].values.reshape(1, -1))

price_normalized_test = normalizer.transform(project_data_test['price'].values.reshape(1, -1))
#reshaping again after normalization

price_normalized_train = price_normalized_train.reshape(-1, 1)
```

```
price_normalized_test =  price_normalized_test.
reshape(-1, 1)

print('After normalization')
print(price_normalized_train.shape)

print(price_normalized_test.shape)
```

```
After normalization
(20100, 1)
(9900, 1)
```

In [58]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['quantity'].v
alues.reshape(1,-1))



quantity_normalized_train = normalizer.transfor
m(project_data_train['quantity'].values.reshape
(1, -1))

quantity_normalized_test = normalizer.transform
(project_data_test['quantity'].values.reshape(1
, -1))

#reshaping again after normalization

quantity_normalized_train = quantity_normalized
_train.reshape(-1,1)
quantity_normalized_test = quantity_normalized_
test.reshape(-1,1)

print('After normalization')
print(quantity_normalized_train.shape)

print(quantity_normalized_test.shape)
```

```
After normalization
(20100, 1)
(9900, 1)
```

In [59]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))



previously_posted_projects_normalized_train = normalizer.transform(project_data_train['teacher_number_of_previously_posted_projects'].values.reshape(1, -1))

previously_posted_projects_normalized_test = normalizer.transform(project_data_test['teacher_number_of_previously_posted_projects'].values.reshape(1, -1))

#reshaping again after normalization

previously_posted_projects_normalized_train = previously_posted_projects_normalized_train.reshape(-1,1)
previously_posted_projects_normalized_test = previously_posted_projects_normalized_test.reshape(-1,1)

print('After normalization')
print(previously_posted_projects_normalized_train.shape)

print(previously_posted_projects_normalized_test.shape)
```

```
After normalization
(20100, 1)
(9900, 1)
```

In [60]:

```python
normalizer = Normalizer()
normalizer.fit(project_data_train['essay_word_count'].values.reshape(-1,1))



essay_word_count_normalized_train = normalizer.transform(project_data_train['essay_word_count'].values.reshape(1, -1))

essay_word_count_normalized_test = normalizer.transform(project_data_test['essay_word_count'].values.reshape(1, -1))

#reshaping again after normalization

essay_word_count_normalized_train = essay_word_count_normalized_train.reshape(-1, 1)
essay_word_count_normalized_test = essay_word_count_normalized_test.reshape(-1, 1)

print('After normalization')
print(essay_word_count_normalized_train.shape)

print(essay_word_count_normalized_test.shape)
```

```
After normalization
(20100, 1)
(9900, 1)
```

In [61]:

```python
normalizer = Normalizer()
normalizer.fit(project_data_train['title_word_c
```

```
ount'].values.reshape(-1,1))


title_word_count_normalized_train = normalizer.
transform(project_data_train['title_word_count'
].values.reshape(1, -1))

title_word_count_normalized_test = normalizer.t
ransform(project_data_test['title_word_count'].
values.reshape(1, -1))

#reshaping again after normalization

title_word_count_normalized_train = title_word_
count_normalized_train.reshape(-1, 1)
title_word_count_normalized_test = title_word_c
ount_normalized_test.reshape(-1, 1)




print('After normalization')
print(title_word_count_normalized_train.shape)

print(title_word_count_normalized_test.shape)
```

```
After normalization

(20100, 1)
(9900, 1)
```

In [62]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['neg'].values
.reshape(-1,1))
```

```
sent_neg_train = normalizer.transform(project_data_train['neg'].values.reshape(1, -1))

sent_neg_test = normalizer.transform(project_data_test['neg'].values.reshape(1, -1))

#reshaping again after normalization
sent_neg_train = sent_neg_train.reshape(-1,1)
sent_neg_test = sent_neg_test.reshape(-1,1)



print('After normalization')
print(sent_neg_train.shape)

print(sent_neg_test.shape)
```

```
After normalization
(20100, 1)
(9900, 1)
```

In [63]:
```
normalizer = Normalizer()
normalizer.fit(project_data_train['pos'].values.reshape(-1,1))


sent_pos_train = normalizer.transform(project_data_train['pos'].values.reshape(1, -1))

sent_pos_test = normalizer.transform(project_data_test['pos'].values.reshape(1, -1))

#reshaping again after normalization
sent_pos_train = sent_pos_train.reshape(-1,1)
sent_pos_test = sent_pos_test.reshape(-1,1)
```

```python


print('After normalization')

print(sent_pos_train.shape)

print(sent_pos_test.shape)
```

```
After normalization
(20100, 1)
(9900, 1)
```

In [64]:

```python
normalizer = Normalizer()
normalizer.fit(project_data_train['neu'].values
.reshape(-1,1))


sent_neu_train = normalizer.transform(project_d
ata_train['neu'].values.reshape(1, -1))

sent_neu_test = normalizer.transform(project_da
ta_test['neu'].values.reshape(1, -1))

#reshaping again after normalization
sent_neu_train = sent_neu_train.reshape(-1,1)
sent_neu_test = sent_neu_test.reshape(-1,1)


print('After normalization')
print(sent_neu_train.shape)

print(sent_neu_test.shape)
```

```
After normalization
(20100, 1)
(9900, 1)
```

In [65]:
```
normalizer = Normalizer()
normalizer.fit(project_data_train['compound'].v
alues.reshape(-1,1))


sent_compound_train = normalizer.transform(proj
ect_data_train['compound'].values.reshape(1, -1
))

sent_compound_test = normalizer.transform(proje
ct_data_test['compound'].values.reshape(1, -1))


#reshaping again after normalization
sent_compound_train = sent_compound_train.resha
pe(-1,1)
sent_compound_test = sent_compound_test.reshape
(-1,1)


print('After normalization')
print(sent_compound_train.shape)

print(sent_compound_test.shape)
```

```
After normalization
(20100, 1)
(9900, 1)
```

Response coding for Categorical Data

In [0]:
```
#https://stackoverflow.com/questions/11869910/p
```

```
andas-filter-rows-of-dataframe-with-operator-ch
aining
def mask(df, key, value):
    return df[df[key] == value]

def get_response(data,data_label):
    cat_values = np.unique(data).tolist()
    df = pd.DataFrame({'feature':data.values.to
list(),'label':data_label.values.tolist()})
    pd.DataFrame.mask = mask

    accep = {};reject={};prob_neg = {};prob_pos
={}
    for i in cat_values:
        count_0 = len(df.mask('feature', i).mas
k('label', 0))
        count_1 = len(df.mask('feature', i).mas
k('label', 1))
        total   =   count_0 + count_1
        prob_0 = count_0/total
        prob_1 = count_1/total
        accep[i] = count_1
        reject[i] = count_0
        prob_neg[i] = prob_0
        prob_pos[i] = prob_1

    return prob_neg,prob_pos
```

In [0]:

```
cat_0_train = get_response(project_data_train[
'clean_categories'],y_train)[0]
cat_1_train = get_response(project_data_train[
'clean_categories'],y_train)[1]
```

In [0]:

```
subcat_0_train = get_response(project_data_trai
```

```
n['clean_subcategories'],y_train)[0]
subcat_1_train = get_response(project_data_trai
n['clean_subcategories'],y_train)[1]
```

In [0]:

```
state_0_train = get_response(project_data_train
['school_state'],y_train)[0]
state_1_train = get_response(project_data_train
['school_state'],y_train)[1]
```

In [0]:

```
prefix_0_train = get_response(project_data_trai
n['teacher_prefix'],y_train)[0]
prefix_1_train = get_response(project_data_trai
n['teacher_prefix'],y_train)[1]
```

In [0]:

```
grad_cat_0_train = get_response(project_data_tr
ain['project_grade_category'],y_train)[0]
grad_cat_1_train = get_response(project_data_tr
ain['project_grade_category'],y_train)[1]
```

In [0]:

```
cat_0_test = get_response(project_data_test['cl
ean_categories'],y_test)[0]
cat_1_test = get_response(project_data_test['cl
ean_categories'],y_test)[1]
```

In [0]:

```
subcat_0_test = get_response(project_data_test[
'clean_subcategories'],y_test)[0]
subcat_1_test = get_response(project_data_test[
'clean_subcategories'],y_test)[1]
```

In [0]:

```
state_0_test = get_response(project_data_test[
'school_state'],y_test)[0]
state_1_test = get_response(project_data_test[
'school_state'],y_test)[1]
```

In [0]:
```
prefix_0_test = get_response(project_data_test[
'teacher_prefix'],y_test)[0]
prefix_1_test = get_response(project_data_test[
'teacher_prefix'],y_test)[1]
```

In [0]:
```
grad_cat_0_test = get_response(project_data_tes
t['project_grade_category'],y_test)[0]
grad_cat_1_test = get_response(project_data_tes
t['project_grade_category'],y_test)[1]
```

In [77]:
```
cat_0_train
```

Out[77]:
```
{'AppliedLearning': 0.17590027700831026,
 'AppliedLearning Health_Sports': 0.16326
530612244897,
 'AppliedLearning History_Civics': 0.2307
6923076923078,
 'AppliedLearning Literacy_Language': 0.1
4354066985645933,
 'AppliedLearning Math_Science': 0.189189
1891891892,
 'AppliedLearning Music_Arts': 0.19580419
58041958,
 'AppliedLearning SpecialNeeds': 0.169230
76923076924,
 'AppliedLearning Warmth Care_Hunger': 0.
0,
 'Health_Sports': 0.15204981837052414,
 'Health_Sports AppliedLearning': 0.21153
846153846154,
 'Health_Sports History_Civics': 0.0,
 'Health_Sports Literacy_Language': 0.173
6111111111111,
 'Health_Sports Math_Science': 0.125,
 'Health_Sports Music_Arts': 0.3448275862
```

068966,
 'Health_Sports SpecialNeeds': 0.14516129
032258066,
 'Health_Sports Warmth Care_Hunger': 0.16
666666666666666,
 'History_Civics': 0.15542521994134897,
 'History_Civics AppliedLearning': 0.1333
3333333333333,
 'History_Civics Health_Sports': 0.0,
 'History_Civics Literacy_Language': 0.09
195402298850575,
 'History_Civics Math_Science': 0.1212121
2121212122,
 'History_Civics Music_Arts': 0.169811320
75471697,
 'History_Civics SpecialNeeds': 0.2121212
1212121213,
 'Literacy_Language': 0.1331652532661013
2,
 'Literacy_Language AppliedLearning': 0.1
3978494623655913,
 'Literacy_Language Health_Sports': 0.272
7272727272727,
 'Literacy_Language History_Civics': 0.15
827338129496402,
 'Literacy_Language Math_Science': 0.1334
327245620574,
 'Literacy_Language Music_Arts': 0.161290
32258064516,
 'Literacy_Language SpecialNeeds': 0.1537
3563218390804,
 'Literacy_Language Warmth Care_Hunger':
0.0,
 'Math_Science': 0.18064098413726126,
 'Math_Science AppliedLearning': 0.16,
 'Math_Science Health_Sports': 0.23188405

```
79710145,
 'Math_Science History_Civics': 0.1826086
956521739,
 'Math_Science Literacy_Language': 0.1029
082774049217,
 'Math_Science Music_Arts': 0.19218241042
345277,
 'Math_Science SpecialNeeds': 0.203296703
2967033,
 'Math_Science Warmth Care_Hunger': 0.333
3333333333333,
 'Music_Arts': 0.15602094240837697,
 'Music_Arts AppliedLearning': 0.0,
 'Music_Arts Health_Sports': 0.5,
 'Music_Arts History_Civics': 0.5,
 'Music_Arts SpecialNeeds': 0.14285714285
714285,
 'Music_Arts Warmth Care_Hunger': 1.0,
 'SpecialNeeds': 0.20539152759948653,
 'SpecialNeeds Health_Sports': 0.22222222
22222222,
 'SpecialNeeds Music_Arts': 0.17307692307
692307,
 'SpecialNeeds Warmth Care_Hunger': 0.0,
 'Warmth Care_Hunger': 0.0661157024793388
4}
```

In [78]:
```
cat_1_train
```

Out[78]:
```
{'AppliedLearning': 0.8240997229916898,
 'AppliedLearning Health_Sports': 0.83673
46938775511,
 'AppliedLearning History_Civics': 0.7692
307692307693,
 'AppliedLearning Literacy_Language': 0.8
564593301435407,
```

'AppliedLearning Math_Science': 0.810810
8108108109,
'AppliedLearning Music_Arts': 0.80419580
41958042,
'AppliedLearning SpecialNeeds': 0.830769
2307692308,
'AppliedLearning Warmth Care_Hunger': 1.
0,
'Health_Sports': 0.8479501816294759,
'Health_Sports AppliedLearning': 0.78846
15384615384,
'Health_Sports History_Civics': 1.0,
'Health_Sports Literacy_Language': 0.826
3888888888888,
'Health_Sports Math_Science': 0.875,
'Health_Sports Music_Arts': 0.6551724137
931034,
'Health_Sports SpecialNeeds': 0.85483870
96774194,
'Health_Sports Warmth Care_Hunger': 0.83
33333333333334,
'History_Civics': 0.844574780058651,
'History_Civics AppliedLearning': 0.8666
666666666667,
'History_Civics Health_Sports': 1.0,
'History_Civics Literacy_Language': 0.90
80459770114943,
'History_Civics Math_Science': 0.8787878
787878788,
'History_Civics Music_Arts': 0.830188679
2452831,
'History_Civics SpecialNeeds': 0.7878787
878787878,
'Literacy_Language': 0.8668347467338987,
'Literacy_Language AppliedLearning': 0.8
602150537634409,

'Literacy_Language Health_Sports': 0.7272727272727273,
'Literacy_Language History_Civics': 0.841726618705036,
'Literacy_Language Math_Science': 0.8665672754379427,
'Literacy_Language Music_Arts': 0.8387096774193549,
'Literacy_Language SpecialNeeds': 0.8462643678160919,
'Literacy_Language Warmth Care_Hunger': 1.0,
'Math_Science': 0.8193590158627387,
'Math_Science AppliedLearning': 0.84,
'Math_Science Health_Sports': 0.7681159420289855,
'Math_Science History_Civics': 0.8173913043478261,
'Math_Science Literacy_Language': 0.8970917225950783,
'Math_Science Music_Arts': 0.8078175895765473,
'Math_Science SpecialNeeds': 0.7967032967032966,
'Math_Science Warmth Care_Hunger': 0.6666666666666666,
'Music_Arts': 0.8439790575916231,
'Music_Arts AppliedLearning': 1.0,
'Music_Arts Health_Sports': 0.5,
'Music_Arts History_Civics': 0.5,
'Music_Arts SpecialNeeds': 0.8571428571428571,
'Music_Arts Warmth Care_Hunger': 0.0,
'SpecialNeeds': 0.7946084724005135,
'SpecialNeeds Health_Sports': 0.7777777777777778,

```
 'SpecialNeeds Music_Arts': 0.82692307692
30769,
 'SpecialNeeds Warmth Care_Hunger': 1.0,
 'Warmth Care_Hunger': 0.933884297520661
2}
```

In [0]:

```
cat_neg_train = []
cat_pos_train = []
for i in project_data_train['clean_categories'
]:
    cat_neg_train.append(cat_0_train[i])
    cat_pos_train.append(cat_1_train[i])
project_data_train['cat_0'] = cat_neg_train
project_data_train['cat_1'] = cat_pos_train
```

In [80]:

```
subcat_0_train
```

Out[80]:

```
{'AppliedSciences': 0.18816067653276955,
 'AppliedSciences CharacterEducation': 0.
5,
 'AppliedSciences Civics_Government': 0.
0,
 'AppliedSciences College_CareerPrep': 0.
18181818181818182,
 'AppliedSciences CommunityService': 0.0,
 'AppliedSciences ESL': 0.055555555555555
55,
 'AppliedSciences EarlyDevelopment': 0.05
714285714285714,
 'AppliedSciences Economics': 1.0,
 'AppliedSciences EnvironmentalScience':
0.2023121387283237,
 'AppliedSciences Extracurricular': 0.0,
 'AppliedSciences FinancialLiteracy': 1.
0,
```

'AppliedSciences Gym_Fitness': 0.5,
'AppliedSciences Health_LifeScience': 0.
16346153846153846,
'AppliedSciences Health_Wellness': 0.0,
'AppliedSciences History_Geography': 0.2
3529411764705882,
'AppliedSciences Literacy': 0.1111111111
111111,
'AppliedSciences Literature_Writing': 0.
10810810810810811,
'AppliedSciences Mathematics': 0.1637239
165329053,
'AppliedSciences Music': 0.1428571428571
4285,
'AppliedSciences NutritionEducation': 0.
0,
'AppliedSciences Other': 0.2380952380952
3808,
'AppliedSciences ParentInvolvement': 0.
2,
'AppliedSciences PerformingArts': 0.0,
'AppliedSciences SocialSciences': 0.1111
111111111111,
'AppliedSciences SpecialNeeds': 0.123287
6712328767,
'AppliedSciences TeamSports': 0.0,
'AppliedSciences VisualArts': 0.19117647
058823528,
'CharacterEducation': 0.1228070175438596
4,
'CharacterEducation Civics_Government':
0.0,
'CharacterEducation College_CareerPrep':
0.0625,
'CharacterEducation CommunityService':
0.125,

 'CharacterEducation EarlyDevelopment':
0.38461538461538464,
 'CharacterEducation Economics': 1.0,
 'CharacterEducation EnvironmentalScienc
e': 0.0,
 'CharacterEducation Extracurricular': 0.
18181818181818182,
 'CharacterEducation ForeignLanguages':
0.0,
 'CharacterEducation Gym_Fitness': 0.3333
333333333333,
 'CharacterEducation Health_LifeScience':
0.16666666666666666,
 'CharacterEducation Health_Wellness': 0.
23809523809523808,
 'CharacterEducation History_Geography':
0.0,
 'CharacterEducation Literacy': 0.0895522
3880597014,
 'CharacterEducation Literature_Writing':
0.20689655172413793,
 'CharacterEducation Mathematics': 0.1052
6315789473684,
 'CharacterEducation Music': 0.2,
 'CharacterEducation NutritionEducation':
0.0,
 'CharacterEducation Other': 0.1379310344
8275862,
 'CharacterEducation ParentInvolvement':
0.3333333333333333,
 'CharacterEducation PerformingArts': 0.
5,
 'CharacterEducation SocialSciences': 0.
2,
 'CharacterEducation SpecialNeeds': 0.235
29411764705882,

'CharacterEducation TeamSports': 0.66666
66666666666,
'CharacterEducation VisualArts': 0.2,
'Civics_Government': 0.0,
'Civics_Government College_CareerPrep':
0.0,
'Civics_Government CommunityService': 0.
0,
'Civics_Government ESL': 0.0,
'Civics_Government Economics': 0.0,
'Civics_Government EnvironmentalScienc
e': 1.0,
'Civics_Government Extracurricular': 1.
0,
'Civics_Government FinancialLiteracy':
0.16666666666666666,
'Civics_Government Health_LifeScience':
0.25,
'Civics_Government Health_Wellness': 0.
0,
'Civics_Government History_Geography':
0.15217391304347827,
'Civics_Government Literacy': 0.06896551
724137931,
'Civics_Government Literature_Writing':
0.2,
'Civics_Government Mathematics': 0.0,
'Civics_Government SocialSciences': 0.0,
'Civics_Government SpecialNeeds': 0.5,
'Civics_Government TeamSports': 0.0,
'Civics_Government VisualArts': 0.0,
'College_CareerPrep': 0.2297297297297297
4,
'College_CareerPrep CommunityService':
0.2,
'College_CareerPrep EarlyDevelopment':

0.0,
 'College_CareerPrep Economics': 0.0,
 'College_CareerPrep EnvironmentalScienc
e': 0.25,
 'College_CareerPrep Extracurricular': 0.
0,
 'College_CareerPrep FinancialLiteracy':
0.4,
 'College_CareerPrep ForeignLanguages':
0.25,
 'College_CareerPrep Gym_Fitness': 1.0,
 'College_CareerPrep Health_LifeScience':
0.0,
 'College_CareerPrep Health_Wellness': 0.
3333333333333333,
 'College_CareerPrep History_Geography':
0.3333333333333333,
 'College_CareerPrep Literacy': 0.0697674
4186046512,
 'College_CareerPrep Literature_Writing':
0.14754098360655737,
 'College_CareerPrep Mathematics': 0.2413
793103448276,
 'College_CareerPrep NutritionEducation':
0.0,
 'College_CareerPrep Other': 0.3636363636
3636365,
 'College_CareerPrep ParentInvolvement':
0.4,
 'College_CareerPrep PerformingArts': 0.2
5,
 'College_CareerPrep SocialSciences': 0.1
4285714285714285,
 'College_CareerPrep SpecialNeeds': 0.25,
 'College_CareerPrep TeamSports': 0.0,
 'College_CareerPrep VisualArts': 0.04761

9047619047616,
 'College_CareerPrep Warmth Care_Hunger': 0.0,
 'CommunityService': 0.25,
 'CommunityService Economics': 0.5,
 'CommunityService EnvironmentalScience': 0.3333333333333333,
 'CommunityService Extracurricular': 0.0,
 'CommunityService Gym_Fitness': 0.0,
 'CommunityService Health_LifeScience': 0.0,
 'CommunityService Health_Wellness': 0.4,
 'CommunityService Literacy': 0.2,
 'CommunityService Literature_Writing': 0.25,
 'CommunityService Mathematics': 0.25,
 'CommunityService NutritionEducation': 1.0,
 'CommunityService Other': 0.0,
 'CommunityService ParentInvolvement': 0.0,
 'CommunityService PerformingArts': 0.0,
 'CommunityService SocialSciences': 0.5,
 'CommunityService SpecialNeeds': 0.0,
 'CommunityService VisualArts': 0.2727272727272727,
 'ESL': 0.1744186046511628,
 'ESL EarlyDevelopment': 0.08333333333333333,
 'ESL EnvironmentalScience': 0.4,
 'ESL ForeignLanguages': 0.2,
 'ESL Health_LifeScience': 0.3333333333333333,
 'ESL Health_Wellness': 0.3333333333333333,
 'ESL History_Geography': 0.2857142857142

857,
 'ESL Literacy': 0.13981042654028436,
 'ESL Literature_Writing': 0.173913043478
26086,
 'ESL Mathematics': 0.2,
 'ESL Other': 0.0,
 'ESL ParentInvolvement': 0.0,
 'ESL PerformingArts': 0.5,
 'ESL SocialSciences': 0.0,
 'ESL SpecialNeeds': 0.1724137931034483,
 'ESL VisualArts': 0.2,
 'EarlyDevelopment': 0.15476190476190477,
 'EarlyDevelopment EnvironmentalScience':
0.2857142857142857,
 'EarlyDevelopment Extracurricular': 0.0,
 'EarlyDevelopment FinancialLiteracy': 0.
0,
 'EarlyDevelopment Gym_Fitness': 0.0,
 'EarlyDevelopment Health_LifeScience':
0.14285714285714285,
 'EarlyDevelopment Health_Wellness': 0.06
666666666666667,
 'EarlyDevelopment History_Geography': 0.
0,
 'EarlyDevelopment Literacy': 0.162962962
96296298,
 'EarlyDevelopment Literature_Writing':
0.1276595744680851,
 'EarlyDevelopment Mathematics': 0.180555
55555555555,
 'EarlyDevelopment Music': 0.25,
 'EarlyDevelopment NutritionEducation':
0.0,
 'EarlyDevelopment Other': 0.035714285714
28571,
 'EarlyDevelopment ParentInvolvement': 0.

1,
 'EarlyDevelopment PerformingArts': 0.0,
 'EarlyDevelopment SocialSciences': 0.0,
 'EarlyDevelopment SpecialNeeds': 0.13793
103448275862,
 'EarlyDevelopment VisualArts': 0.3225806
451612903,
 'EarlyDevelopment Warmth Care_Hunger':
0.0,
 'Economics': 0.1111111111111111,
 'Economics FinancialLiteracy': 0.3125,
 'Economics History_Geography': 0.0,
 'Economics Literacy': 0.0,
 'Economics Mathematics': 0.0,
 'Economics Other': 0.0,
 'Economics SpecialNeeds': 1.0,
 'EnvironmentalScience': 0.17209302325581
396,
 'EnvironmentalScience Extracurricular':
0.0,
 'EnvironmentalScience FinancialLiterac
y': 0.0,
 'EnvironmentalScience ForeignLanguages':
0.0,
 'EnvironmentalScience Health_LifeScienc
e': 0.22085889570552147,
 'EnvironmentalScience Health_Wellness':
0.2857142857142857,
 'EnvironmentalScience History_Geograph
y': 0.16129032258064516,
 'EnvironmentalScience Literacy': 0.17073
170731707318,
 'EnvironmentalScience Literature_Writin
g': 0.03278688524590164,
 'EnvironmentalScience Mathematics': 0.14
960629921259844,

'EnvironmentalScience Music': 0.0,
'EnvironmentalScience NutritionEducatio
n': 0.7142857142857143,
'EnvironmentalScience Other': 0.0,
'EnvironmentalScience ParentInvolvemen
t': 0.0,
'EnvironmentalScience PerformingArts':
1.0,
'EnvironmentalScience SocialSciences':
0.07692307692307693,
'EnvironmentalScience SpecialNeeds': 0.1
6666666666666666,
'EnvironmentalScience TeamSports': 0.0,
'EnvironmentalScience VisualArts': 0.181
81818181818182,
'EnvironmentalScience Warmth Care_Hunge
r': 0.0,
'Extracurricular': 0.17647058823529413,
'Extracurricular Health_LifeScience': 0.
0,
'Extracurricular Health_Wellness': 0.0,
'Extracurricular History_Geography': 0.
0,
'Extracurricular Literacy': 0.3333333333
333333,
'Extracurricular Literature_Writing': 0.
0,
'Extracurricular Mathematics': 0.0833333
3333333333,
'Extracurricular Music': 0.5,
'Extracurricular Other': 0.3,
'Extracurricular ParentInvolvement': 0.
0,
'Extracurricular PerformingArts': 0.0,
'Extracurricular SocialSciences': 0.0,
'Extracurricular SpecialNeeds': 1.0,

 'Extracurricular TeamSports': 0.0,
 'Extracurricular VisualArts': 0.0,
 'FinancialLiteracy': 0.09375,
 'FinancialLiteracy History_Geography':
0.0,
 'FinancialLiteracy Literacy': 0.33333333
33333333,
 'FinancialLiteracy Literature_Writing':
0.0,
 'FinancialLiteracy Mathematics': 0.15384
615384615385,
 'FinancialLiteracy Other': 0.0,
 'FinancialLiteracy SpecialNeeds': 0.1428
5714285714285,
 'ForeignLanguages': 0.13114754098360656,
 'ForeignLanguages Health_LifeScience':
0.0,
 'ForeignLanguages Health_Wellness': 0.0,
 'ForeignLanguages History_Geography': 0.
0,
 'ForeignLanguages Literacy': 0.173076923
07692307,
 'ForeignLanguages Literature_Writing':
0.3076923076923077,
 'ForeignLanguages Mathematics': 0.0,
 'ForeignLanguages Music': 0.0,
 'ForeignLanguages PerformingArts': 0.0,
 'ForeignLanguages SpecialNeeds': 0.0,
 'ForeignLanguages VisualArts': 0.3333333
333333333,
 'Gym_Fitness': 0.15315315315315314,
 'Gym_Fitness Health_LifeScience': 0.0,
 'Gym_Fitness Health_Wellness': 0.1176470
5882352941,
 'Gym_Fitness History_Geography': 0.0,
 'Gym_Fitness Literacy': 0.625,

```
 'Gym_Fitness Literature_Writing': 0.0,
 'Gym_Fitness Mathematics': 0.22222222222
22222,
 'Gym_Fitness Music': 0.25,
 'Gym_Fitness NutritionEducation': 0.2142
8571428571427,
 'Gym_Fitness Other': 1.0,
 'Gym_Fitness ParentInvolvement': 0.0,
 'Gym_Fitness PerformingArts': 0.33333333
33333333,
 'Gym_Fitness SpecialNeeds': 0.1851851851
8518517,
 'Gym_Fitness TeamSports': 0.2,
 'Gym_Fitness VisualArts': 0.333333333333
3333,
 'Health_LifeScience': 0.164383561643835
6,
 'Health_LifeScience Health_Wellness': 0.
17857142857142858,
 'Health_LifeScience History_Geography':
0.1,
 'Health_LifeScience Literacy': 0.0625,
 'Health_LifeScience Literature_Writing':
0.10810810810810811,
 'Health_LifeScience Mathematics': 0.1919
191919191919,
 'Health_LifeScience Music': 0.0,
 'Health_LifeScience NutritionEducation':
0.14285714285714285,
 'Health_LifeScience Other': 0.5,
 'Health_LifeScience ParentInvolvement':
0.0,
 'Health_LifeScience SocialSciences': 0.2
5,
 'Health_LifeScience SpecialNeeds': 0.24,
 'Health_LifeScience VisualArts': 0.21428
```

571428571427,
 'Health_Wellness': 0.15453194650817237,
 'Health_Wellness History_Geography': 0.
0,
 'Health_Wellness Literacy': 0.1704545454
5454544,
 'Health_Wellness Literature_Writing': 0.
10869565217391304,
 'Health_Wellness Mathematics': 0.1071428
5714285714,
 'Health_Wellness Music': 0.25,
 'Health_Wellness NutritionEducation': 0.
13043478260869565,
 'Health_Wellness Other': 0.1860465116279
0697,
 'Health_Wellness ParentInvolvement': 0.
5,
 'Health_Wellness PerformingArts': 0.25,
 'Health_Wellness SocialSciences': 0.0,
 'Health_Wellness SpecialNeeds': 0.134883
72093023257,
 'Health_Wellness TeamSports': 0.17910447
76119403,
 'Health_Wellness VisualArts': 0.42857142
857142855,
 'Health_Wellness Warmth Care_Hunger': 0.
16666666666666666,
 'History_Geography': 0.2075471698113207
6,
 'History_Geography Literacy': 0.06862745
098039216,
 'History_Geography Literature_Writing':
0.10185185185185185,
 'History_Geography Mathematics': 0.08,
 'History_Geography Music': 0.16666666666
666666,

'History_Geography Other': 0.25,
'History_Geography PerformingArts': 0.2
5,
'History_Geography SocialSciences': 0.13
793103448275862,
'History_Geography SpecialNeeds': 0.2,
'History_Geography VisualArts': 0.162162
16216216217,
'Literacy': 0.12570145903479238,
'Literacy Literature_Writing': 0.1288404
360753221,
'Literacy Mathematics': 0.12922077922077
92,
'Literacy Music': 0.041666666666666664,
'Literacy NutritionEducation': 0.5,
'Literacy Other': 0.12,
'Literacy ParentInvolvement': 0.09375,
'Literacy PerformingArts': 0.11111111111
11111,
'Literacy SocialSciences': 0.14925373134
328357,
'Literacy SpecialNeeds': 0.1334841628959
276,
'Literacy TeamSports': 0.333333333333333
3,
'Literacy VisualArts': 0.144329896907216
48,
'Literacy Warmth Care_Hunger': 0.0,
'Literature_Writing': 0.1345911949685534
6,
'Literature_Writing Mathematics': 0.1339
622641509434,
'Literature_Writing Music': 0.125,
'Literature_Writing Other': 0.3333333333
333333,
'Literature_Writing ParentInvolvement':

0.2222222222222222,
 'Literature_Writing PerformingArts': 0.1
875,
 'Literature_Writing SocialSciences': 0.1
6129032258064516,
 'Literature_Writing SpecialNeeds': 0.191
96428571428573,
 'Literature_Writing TeamSports': 0.0,
 'Literature_Writing VisualArts': 0.1875,
 'Literature_Writing Warmth Care_Hunger':
0.0,
 'Mathematics': 0.18633540372670807,
 'Mathematics Music': 0.1111111111111111,
 'Mathematics NutritionEducation': 0.0,
 'Mathematics Other': 0.2,
 'Mathematics ParentInvolvement': 0.16666
666666666666,
 'Mathematics PerformingArts': 0.33333333
33333333,
 'Mathematics SocialSciences': 0.25,
 'Mathematics SpecialNeeds': 0.2304347826
0869565,
 'Mathematics TeamSports': 0.5,
 'Mathematics VisualArts': 0.212765957446
8085,
 'Mathematics Warmth Care_Hunger': 0.5,
 'Music': 0.11742424242424243,
 'Music Other': 0.0,
 'Music ParentInvolvement': 0.0,
 'Music PerformingArts': 0.09580838323353
294,
 'Music SocialSciences': 0.5,
 'Music SpecialNeeds': 0.0588235294117647
05,
 'Music TeamSports': 0.5,
 'Music VisualArts': 0.125,

```
 'NutritionEducation': 0.272727272727272
7,
 'NutritionEducation Other': 0.2,
 'NutritionEducation SocialSciences': 0.
0,
 'NutritionEducation SpecialNeeds': 0.333
3333333333333,
 'NutritionEducation TeamSports': 0.0,
 'Other': 0.1695906432748538,
 'Other ParentInvolvement': 0.25,
 'Other PerformingArts': 1.0,
 'Other SocialSciences': 0.0,
 'Other SpecialNeeds': 0.1698113207547169
7,
 'Other TeamSports': 0.0,
 'Other VisualArts': 0.17647058823529413,
 'ParentInvolvement': 0.25,
 'ParentInvolvement PerformingArts': 0.0,
 'ParentInvolvement SocialSciences': 0.33
33333333333333,
 'ParentInvolvement SpecialNeeds': 0.25,
 'ParentInvolvement VisualArts': 0.333333
3333333333,
 'ParentInvolvement Warmth Care_Hunger':
0.0,
 'PerformingArts': 0.12658227848101267,
 'PerformingArts SocialSciences': 0.5,
 'PerformingArts SpecialNeeds': 0.5,
 'PerformingArts TeamSports': 0.5,
 'PerformingArts VisualArts': 0.318181818
1818182,
 'SocialSciences': 0.1875,
 'SocialSciences SpecialNeeds': 0.125,
 'SocialSciences VisualArts': 0.2,
 'SpecialNeeds': 0.20539152759948653,
 'SpecialNeeds TeamSports': 0.22222222222
```

```
22222,
 'SpecialNeeds VisualArts': 0.17307692307
692307,
 'SpecialNeeds Warmth Care_Hunger': 0.0,
 'TeamSports': 0.15873015873015872,
 'TeamSports VisualArts': 1.0,
 'VisualArts': 0.20240963855421687,
 'VisualArts Warmth Care_Hunger': 1.0,
 'Warmth Care_Hunger': 0.0661157024793388
4}
```

In [81]:

```
subcat_1_train
```

Out[81]:

```
{'AppliedSciences': 0.8118393234672304,
 'AppliedSciences CharacterEducation': 0.
5,
 'AppliedSciences Civics_Government': 1.
0,
 'AppliedSciences College_CareerPrep': 0.
8181818181818182,
 'AppliedSciences CommunityService': 1.0,
 'AppliedSciences ESL': 0.94444444444444444
4,
 'AppliedSciences EarlyDevelopment': 0.94
28571428571428,
 'AppliedSciences Economics': 0.0,
 'AppliedSciences EnvironmentalScience':
0.7976878612716763,
 'AppliedSciences Extracurricular': 1.0,
 'AppliedSciences FinancialLiteracy': 0.
0,
 'AppliedSciences Gym_Fitness': 0.5,
 'AppliedSciences Health_LifeScience': 0.
8365384615384616,
 'AppliedSciences Health_Wellness': 1.0,
 'AppliedSciences History_Geography': 0.7
```

647058823529411,
'AppliedSciences Literacy': 0.8888888888
888888,
'AppliedSciences Literature_Writing': 0.
891891891891891 9,
'AppliedSciences Mathematics': 0.8362760
834670947,
'AppliedSciences Music': 0.8571428571428
571,
'AppliedSciences NutritionEducation': 1.
0,
'AppliedSciences Other': 0.7619047619047
619,
'AppliedSciences ParentInvolvement': 0.
8,
'AppliedSciences PerformingArts': 1.0,
'AppliedSciences SocialSciences': 0.8888
888888888888,
'AppliedSciences SpecialNeeds': 0.876712
3287671232,
'AppliedSciences TeamSports': 1.0,
'AppliedSciences VisualArts': 0.80882352
94117647,
'CharacterEducation': 0.877192982456140
3,
'CharacterEducation Civics_Government':
1.0,
'CharacterEducation College_CareerPrep':
0.9375,
'CharacterEducation CommunityService':
0.875,
'CharacterEducation EarlyDevelopment':
0.6153846153846154,
'CharacterEducation Economics': 0.0,
'CharacterEducation EnvironmentalScienc
e': 1.0,

'CharacterEducation Extracurricular': 0.
8181818181818182,
'CharacterEducation ForeignLanguages':
1.0,
'CharacterEducation Gym_Fitness': 0.6666
666666666666,
'CharacterEducation Health_LifeScience':
0.8333333333333334,
'CharacterEducation Health_Wellness': 0.
7619047619047619,
'CharacterEducation History_Geography':
1.0,
'CharacterEducation Literacy': 0.9104477
611940298,
'CharacterEducation Literature_Writing':
0.7931034482758621,
'CharacterEducation Mathematics': 0.8947
368421052632,
'CharacterEducation Music': 0.8,
'CharacterEducation NutritionEducation':
1.0,
'CharacterEducation Other': 0.8620689655
172413,
'CharacterEducation ParentInvolvement':
0.6666666666666666,
'CharacterEducation PerformingArts': 0.
5,
'CharacterEducation SocialSciences': 0.
8,
'CharacterEducation SpecialNeeds': 0.764
7058823529411,
'CharacterEducation TeamSports': 0.33333
33333333333,
'CharacterEducation VisualArts': 0.8,
'Civics_Government': 1.0,
'Civics_Government College_CareerPrep':

1.0,
'Civics_Government CommunityService': 1.
0,
'Civics_Government ESL': 1.0,
'Civics_Government Economics': 1.0,
'Civics_Government EnvironmentalScienc
e': 0.0,
'Civics_Government Extracurricular': 0.
0,
'Civics_Government FinancialLiteracy':
0.8333333333333334,
'Civics_Government Health_LifeScience':
0.75,
'Civics_Government Health_Wellness': 1.
0,
'Civics_Government History_Geography':
0.8478260869565217,
'Civics_Government Literacy': 0.93103448
27586207,
'Civics_Government Literature_Writing':
0.8,
'Civics_Government Mathematics': 1.0,
'Civics_Government SocialSciences': 1.0,
'Civics_Government SpecialNeeds': 0.5,
'Civics_Government TeamSports': 1.0,
'Civics_Government VisualArts': 1.0,
'College_CareerPrep': 0.770270270270270
3,
'College_CareerPrep CommunityService':
0.8,
'College_CareerPrep EarlyDevelopment':
1.0,
'College_CareerPrep Economics': 1.0,
'College_CareerPrep EnvironmentalScienc
e': 0.75,
'College_CareerPrep Extracurricular': 1.

0,
'College_CareerPrep FinancialLiteracy':
0.6,
'College_CareerPrep ForeignLanguages':
0.75,
'College_CareerPrep Gym_Fitness': 0.0,
'College_CareerPrep Health_LifeScience':
1.0,
'College_CareerPrep Health_Wellness': 0.
6666666666666666,
'College_CareerPrep History_Geography':
0.6666666666666666,
'College_CareerPrep Literacy': 0.9302325
581395349,
'College_CareerPrep Literature_Writing':
0.8524590163934426,
'College_CareerPrep Mathematics': 0.7586
206896551724,
'College_CareerPrep NutritionEducation':
1.0,
'College_CareerPrep Other': 0.6363636363
636364,
'College_CareerPrep ParentInvolvement':
0.6,
'College_CareerPrep PerformingArts': 0.7
5,
'College_CareerPrep SocialSciences': 0.8
571428571428571,
'College_CareerPrep SpecialNeeds': 0.75,
'College_CareerPrep TeamSports': 1.0,
'College_CareerPrep VisualArts': 0.95238
09523809523,
'College_CareerPrep Warmth Care_Hunger':
1.0,
'CommunityService': 0.75,
'CommunityService Economics': 0.5,

'CommunityService EnvironmentalScience': 0.6666666666666666,
'CommunityService Extracurricular': 1.0,
'CommunityService Gym_Fitness': 1.0,
'CommunityService Health_LifeScience': 1.0,
'CommunityService Health_Wellness': 0.6,
'CommunityService Literacy': 0.8,
'CommunityService Literature_Writing': 0.75,
'CommunityService Mathematics': 0.75,
'CommunityService NutritionEducation': 0.0,
'CommunityService Other': 1.0,
'CommunityService ParentInvolvement': 1.0,
'CommunityService PerformingArts': 1.0,
'CommunityService SocialSciences': 0.5,
'CommunityService SpecialNeeds': 1.0,
'CommunityService VisualArts': 0.7272727272727273,
'ESL': 0.8255813953488372,
'ESL EarlyDevelopment': 0.9166666666666666,
'ESL EnvironmentalScience': 0.6,
'ESL ForeignLanguages': 0.8,
'ESL Health_LifeScience': 0.6666666666666666,
'ESL Health_Wellness': 0.6666666666666666,
'ESL History_Geography': 0.7142857142857143,
'ESL Literacy': 0.8601895734597157,
'ESL Literature_Writing': 0.8260869565217391,
'ESL Mathematics': 0.8,

 'ESL Other': 1.0,
 'ESL ParentInvolvement': 1.0,
 'ESL PerformingArts': 0.5,
 'ESL SocialSciences': 1.0,
 'ESL SpecialNeeds': 0.8275862068965517,
 'ESL VisualArts': 0.8,
 'EarlyDevelopment': 0.8452380952380952,
 'EarlyDevelopment EnvironmentalScience':
0.7142857142857143,
 'EarlyDevelopment Extracurricular': 1.0,
 'EarlyDevelopment FinancialLiteracy': 1.
0,
 'EarlyDevelopment Gym_Fitness': 1.0,
 'EarlyDevelopment Health_LifeScience':
0.8571428571428571,
 'EarlyDevelopment Health_Wellness': 0.93
33333333333333,
 'EarlyDevelopment History_Geography': 1.
0,
 'EarlyDevelopment Literacy': 0.837037037
037037,
 'EarlyDevelopment Literature_Writing':
0.8723404255319149,
 'EarlyDevelopment Mathematics': 0.819444
4444444444,
 'EarlyDevelopment Music': 0.75,
 'EarlyDevelopment NutritionEducation':
1.0,
 'EarlyDevelopment Other': 0.964285714285
7143,
 'EarlyDevelopment ParentInvolvement': 0.
9,
 'EarlyDevelopment PerformingArts': 1.0,
 'EarlyDevelopment SocialSciences': 1.0,
 'EarlyDevelopment SpecialNeeds': 0.86206
89655172413,

'EarlyDevelopment VisualArts': 0.6774193548387096,
'EarlyDevelopment Warmth Care_Hunger': 1.0,
'Economics': 0.8888888888888888,
'Economics FinancialLiteracy': 0.6875,
'Economics History_Geography': 1.0,
'Economics Literacy': 1.0,
'Economics Mathematics': 1.0,
'Economics Other': 1.0,
'Economics SpecialNeeds': 0.0,
'EnvironmentalScience': 0.827906976744186,
'EnvironmentalScience Extracurricular': 1.0,
'EnvironmentalScience FinancialLiteracy': 1.0,
'EnvironmentalScience ForeignLanguages': 1.0,
'EnvironmentalScience Health_LifeScience': 0.7791411042944786,
'EnvironmentalScience Health_Wellness': 0.7142857142857143,
'EnvironmentalScience History_Geography': 0.8387096774193549,
'EnvironmentalScience Literacy': 0.8292682926829268,
'EnvironmentalScience Literature_Writing': 0.9672131147540983,
'EnvironmentalScience Mathematics': 0.8503937007874016,
'EnvironmentalScience Music': 1.0,
'EnvironmentalScience NutritionEducation': 0.2857142857142857,
'EnvironmentalScience Other': 1.0,
'EnvironmentalScience ParentInvolvemen

t': 1.0,
'EnvironmentalScience PerformingArts':
0.0,
'EnvironmentalScience SocialSciences':
0.9230769230769231,
'EnvironmentalScience SpecialNeeds': 0.8
333333333333334,
'EnvironmentalScience TeamSports': 1.0,
'EnvironmentalScience VisualArts': 0.818
1818181818182,
'EnvironmentalScience Warmth Care_Hunge
r': 1.0,
'Extracurricular': 0.8235294117647058,
'Extracurricular Health_LifeScience': 1.
0,
'Extracurricular Health_Wellness': 1.0,
'Extracurricular History_Geography': 1.
0,
'Extracurricular Literacy': 0.6666666666
666666,
'Extracurricular Literature_Writing': 1.
0,
'Extracurricular Mathematics': 0.9166666
666666666,
'Extracurricular Music': 0.5,
'Extracurricular Other': 0.7,
'Extracurricular ParentInvolvement': 1.
0,
'Extracurricular PerformingArts': 1.0,
'Extracurricular SocialSciences': 1.0,
'Extracurricular SpecialNeeds': 0.0,
'Extracurricular TeamSports': 1.0,
'Extracurricular VisualArts': 1.0,
'FinancialLiteracy': 0.90625,
'FinancialLiteracy History_Geography':
1.0,

'FinancialLiteracy Literacy': 0.66666666
66666666,
'FinancialLiteracy Literature_Writing':
1.0,
'FinancialLiteracy Mathematics': 0.84615
38461538461,
'FinancialLiteracy Other': 1.0,
'FinancialLiteracy SpecialNeeds': 0.8571
428571428571,
'ForeignLanguages': 0.8688524590163934,
'ForeignLanguages Health_LifeScience':
1.0,
'ForeignLanguages Health_Wellness': 1.0,
'ForeignLanguages History_Geography': 1.
0,
'ForeignLanguages Literacy': 0.826923076
9230769,
'ForeignLanguages Literature_Writing':
0.6923076923076923,
'ForeignLanguages Mathematics': 1.0,
'ForeignLanguages Music': 1.0,
'ForeignLanguages PerformingArts': 1.0,
'ForeignLanguages SpecialNeeds': 1.0,
'ForeignLanguages VisualArts': 0.6666666
666666666,
'Gym_Fitness': 0.8468468468468469,
'Gym_Fitness Health_LifeScience': 1.0,
'Gym_Fitness Health_Wellness': 0.8823529
411764706,
'Gym_Fitness History_Geography': 1.0,
'Gym_Fitness Literacy': 0.375,
'Gym_Fitness Literature_Writing': 1.0,
'Gym_Fitness Mathematics': 0.77777777777
77778,
'Gym_Fitness Music': 0.75,
'Gym_Fitness NutritionEducation': 0.7857

142857142857,
 'Gym_Fitness Other': 0.0,
 'Gym_Fitness ParentInvolvement': 1.0,
 'Gym_Fitness PerformingArts': 0.66666666
66666666,
 'Gym_Fitness SpecialNeeds': 0.8148148148
148148,
 'Gym_Fitness TeamSports': 0.8,
 'Gym_Fitness VisualArts': 0.666666666666
6666,
 'Health_LifeScience': 0.835616438356164
4,
 'Health_LifeScience Health_Wellness': 0.
8214285714285714,
 'Health_LifeScience History_Geography':
0.9,
 'Health_LifeScience Literacy': 0.9375,
 'Health_LifeScience Literature_Writing':
0.8918918918918919,
 'Health_LifeScience Mathematics': 0.8080
808080808081,
 'Health_LifeScience Music': 1.0,
 'Health_LifeScience NutritionEducation':
0.8571428571428571,
 'Health_LifeScience Other': 0.5,
 'Health_LifeScience ParentInvolvement':
1.0,
 'Health_LifeScience SocialSciences': 0.7
5,
 'Health_LifeScience SpecialNeeds': 0.76,
 'Health_LifeScience VisualArts': 0.78571
42857142857,
 'Health_Wellness': 0.8454680534918276,
 'Health_Wellness History_Geography': 1.
0,
 'Health_Wellness Literacy': 0.8295454545

454546,
 'Health_Wellness Literature_Writing': 0.
8913043478260869,
 'Health_Wellness Mathematics': 0.8928571
428571429,
 'Health_Wellness Music': 0.75,
 'Health_Wellness NutritionEducation': 0.
8695652173913043,
 'Health_Wellness Other': 0.8139534883720
93,
 'Health_Wellness ParentInvolvement': 0.
5,
 'Health_Wellness PerformingArts': 0.75,
 'Health_Wellness SocialSciences': 1.0,
 'Health_Wellness SpecialNeeds': 0.865116
2790697674,
 'Health_Wellness TeamSports': 0.82089552
23880597,
 'Health_Wellness VisualArts': 0.57142857
14285714,
 'Health_Wellness Warmth Care_Hunger': 0.
8333333333333334,
 'History_Geography': 0.7924528301886793,
 'History_Geography Literacy': 0.93137254
90196079,
 'History_Geography Literature_Writing':
0.8981481481481481,
 'History_Geography Mathematics': 0.92,
 'History_Geography Music': 0.83333333333
33334,
 'History_Geography Other': 0.75,
 'History_Geography PerformingArts': 0.7
5,
 'History_Geography SocialSciences': 0.86
20689655172413,
 'History_Geography SpecialNeeds': 0.8,

'History_Geography VisualArts': 0.837837
8378378378,
'Literacy': 0.8742985409652076,
'Literacy Literature_Writing': 0.8711595
639246779,
'Literacy Mathematics': 0.87077922077922
08,
'Literacy Music': 0.9583333333333334,
'Literacy NutritionEducation': 0.5,
'Literacy Other': 0.88,
'Literacy ParentInvolvement': 0.90625,
'Literacy PerformingArts': 0.88888888888
88888,
'Literacy SocialSciences': 0.85074626865
67164,
'Literacy SpecialNeeds': 0.8665158371040
724,
'Literacy TeamSports': 0.666666666666666
6,
'Literacy VisualArts': 0.855670103092783
5,
'Literacy Warmth Care_Hunger': 1.0,
'Literature_Writing': 0.865408805031446
6,
'Literature_Writing Mathematics': 0.8660
377358490566,
'Literature_Writing Music': 0.875,
'Literature_Writing Other': 0.6666666666
666666,
'Literature_Writing ParentInvolvement':
0.7777777777777778,
'Literature_Writing PerformingArts': 0.8
125,
'Literature_Writing SocialSciences': 0.8
387096774193549,
'Literature_Writing SpecialNeeds': 0.808

0357142857143,
 'Literature_Writing TeamSports': 1.0,
 'Literature_Writing VisualArts': 0.8125,
 'Literature_Writing Warmth Care_Hunger':
1.0,
 'Mathematics': 0.8136645962732919,
 'Mathematics Music': 0.8888888888888888,
 'Mathematics NutritionEducation': 1.0,
 'Mathematics Other': 0.8,
 'Mathematics ParentInvolvement': 0.83333
33333333334,
 'Mathematics PerformingArts': 0.66666666
66666666,
 'Mathematics SocialSciences': 0.75,
 'Mathematics SpecialNeeds': 0.7695652173
913043,
 'Mathematics TeamSports': 0.5,
 'Mathematics VisualArts': 0.787234042553
1915,
 'Mathematics Warmth Care_Hunger': 0.5,
 'Music': 0.8825757575757576,
 'Music Other': 1.0,
 'Music ParentInvolvement': 1.0,
 'Music PerformingArts': 0.90419161676646
71,
 'Music SocialSciences': 0.5,
 'Music SpecialNeeds': 0.941176470588235
3,
 'Music TeamSports': 0.5,
 'Music VisualArts': 0.875,
 'NutritionEducation': 0.727272727272727
3,
 'NutritionEducation Other': 0.8,
 'NutritionEducation SocialSciences': 1.
0,
 'NutritionEducation SpecialNeeds': 0.666

6666666666666,
 'NutritionEducation TeamSports': 1.0,
 'Other': 0.8304093567251462,
 'Other ParentInvolvement': 0.75,
 'Other PerformingArts': 0.0,
 'Other SocialSciences': 1.0,
 'Other SpecialNeeds': 0.830188679245283
1,
 'Other TeamSports': 1.0,
 'Other VisualArts': 0.8235294117647058,
 'ParentInvolvement': 0.75,
 'ParentInvolvement PerformingArts': 1.0,
 'ParentInvolvement SocialSciences': 0.66
66666666666666,
 'ParentInvolvement SpecialNeeds': 0.75,
 'ParentInvolvement VisualArts': 0.666666
6666666666,
 'ParentInvolvement Warmth Care_Hunger':
1.0,
 'PerformingArts': 0.8734177215189873,
 'PerformingArts SocialSciences': 0.5,
 'PerformingArts SpecialNeeds': 0.5,
 'PerformingArts TeamSports': 0.5,
 'PerformingArts VisualArts': 0.681818181
8181818,
 'SocialSciences': 0.8125,
 'SocialSciences SpecialNeeds': 0.875,
 'SocialSciences VisualArts': 0.8,
 'SpecialNeeds': 0.7946084724005135,
 'SpecialNeeds TeamSports': 0.77777777777
77778,
 'SpecialNeeds VisualArts': 0.82692307692
30769,
 'SpecialNeeds Warmth Care_Hunger': 1.0,
 'TeamSports': 0.8412698412698413,
 'TeamSports VisualArts': 0.0,

```
 'VisualArts': 0.7975903614457831,
 'VisualArts Warmth Care_Hunger': 0.0,
 'Warmth Care_Hunger': 0.933884297520661
2}
```

In [0]:

```
subcat_neg_train = []
subcat_pos_train = []
for i in project_data_train['clean_subcategorie
s']:
    subcat_neg_train.append(subcat_0_train[i])
    subcat_pos_train.append(subcat_1_train[i])
project_data_train['subcat_0'] = subcat_neg_tra
in
project_data_train['subcat_1'] = subcat_pos_tra
in
```

In [83]:

```
state_0_train
```

Out[83]:

```
{'AK': 0.20833333333333334,
 'AL': 0.125,
 'AR': 0.20238095238095238,
 'AZ': 0.16279069767441862,
 'CA': 0.14130434782608695,
 'CO': 0.18041237113402062,
 'CT': 0.10869565217391304,
 'DC': 0.24468085106382978,
 'DE': 0.08196721311475409,
 'FL': 0.17353951890034364,
 'GA': 0.14545454545454545,
 'HI': 0.13953488372093023,
 'IA': 0.18110236220472442,
 'ID': 0.18803418803418803,
 'IL': 0.16030534351145037,
 'IN': 0.14870689655172414,
 'KS': 0.12612612612612611,
```

```
 'KY': 0.11016949152542373,
 'LA': 0.17218543046357615,
 'MA': 0.14814814814814814,
 'MD': 0.15789473684210525,
 'ME': 0.1827956989247312,
 'MI': 0.16850393700787403,
 'MN': 0.13488372093023257,
 'MO': 0.13219616204690832,
 'MS': 0.18376068376068377,
 'MT': 0.3111111111111111,
 'NC': 0.14093264248704662,
 'ND': 0.12,
 'NE': 0.12280701754385964,
 'NH': 0.13793103448275862,
 'NJ': 0.17215189873417722,
 'NM': 0.12359550561797752,
 'NV': 0.14942528735632185,
 'NY': 0.13882863340563992,
 'OH': 0.1326530612244898,
 'OK': 0.16,
 'OR': 0.2088888888888889,
 'PA': 0.15544041450777202,
 'RI': 0.15384615384615385,
 'SC': 0.14620689655172414,
 'SD': 0.08695652173913043,
 'TN': 0.15335463258785942,
 'TX': 0.18615040953090098,
 'UT': 0.17405063291139242,
 'VA': 0.16580310880829016,
 'VT': 0.07142857142857142,
 'WA': 0.12641083521444696,
 'WI': 0.16363636363636364,
 'WV': 0.14583333333333334,
 'WY': 0.14814814814814814}
```

In [84]:

```
state_1_train
```

Out[84]:

```
{'AK': 0.7916666666666666,
 'AL': 0.875,
 'AR': 0.7976190476190477,
 'AZ': 0.8372093023255814,
 'CA': 0.8586956521739131,
 'CO': 0.8195876288659794,
 'CT': 0.8913043478260869,
 'DC': 0.7553191489361702,
 'DE': 0.9180327868852459,
 'FL': 0.8264604810996563,
 'GA': 0.8545454545454545,
 'HI': 0.8604651162790697,
 'IA': 0.8188976377952756,
 'ID': 0.811965811965812,
 'IL': 0.8396946564885496,
 'IN': 0.8512931034482759,
 'KS': 0.8738738738738738,
 'KY': 0.8898305084745762,
 'LA': 0.8278145695364238,
 'MA': 0.8518518518518519,
 'MD': 0.8421052631578947,
 'ME': 0.8172043010752689,
 'MI': 0.831496062992126,
 'MN': 0.8651162790697674,
 'MO': 0.8678038379530917,
 'MS': 0.8162393162393162,
 'MT': 0.6888888888888889,
 'NC': 0.8590673575129534,
 'ND': 0.88,
 'NE': 0.8771929824561403,
 'NH': 0.8620689655172413,
 'NJ': 0.8278481012658228,
 'NM': 0.8764044943820225,
 'NV': 0.8505747126436781,
```

```
 'NY': 0.8611713665943601,
 'OH': 0.8673469387755102,
 'OK': 0.84,
 'OR': 0.7911111111111111,
 'PA': 0.844559585492228,
 'RI': 0.8461538461538461,
 'SC': 0.8537931034482759,
 'SD': 0.9130434782608695,
 'TN': 0.8466453674121406,
 'TX': 0.813849590469099,
 'UT': 0.8259493670886076,
 'VA': 0.8341968911917098,
 'VT': 0.9285714285714286,
 'WA': 0.873589164785553,
 'WI': 0.8363636363636363,
 'WV': 0.8541666666666666,
 'WY': 0.8518518518518519}
```

In [0]:

```
state_neg_train = []
state_pos_train = []
for i in project_data_train['school_state']:
    state_neg_train.append(state_0_train[i])
    state_pos_train.append(state_1_train[i])
project_data_train['state_0'] = state_neg_train
project_data_train['state_1'] = state_pos_train
```

In [86]:

```
prefix_0_train
```

Out[86]:

```
{'Mr': 0.15717884130982368,
 'Mrs': 0.15088757396449703,
 'Ms': 0.15403795539548415,
 'Teacher': 0.215311004784689}
```

In [87]:

```
prefix_1_train
```

Out[87]:
```
{'Mr': 0.8428211586901764,
 'Mrs': 0.849112426035503,
 'Ms': 0.8459620446045159,
 'Teacher': 0.784688995215311}
```

In [0]:
```
prefix_neg_train = []
prefix_pos_train = []
for i in project_data_train['teacher_prefix']:
    prefix_neg_train.append(prefix_0_train[i])
    prefix_pos_train.append(prefix_1_train[i])
project_data_train['prefix_0'] = prefix_neg_tra
in
project_data_train['prefix_1'] = prefix_pos_tra
in
```

In [89]:
```
grad_cat_0_train
```

Out[89]:
```
{'Grades_3_5': 0.14500886001181335,
 'Grades_6_8': 0.1611978337050016,
 'Grades_9_12': 0.16675037669512807,
 'Grades_PreK_2': 0.15552573798487435}
```

In [90]:
```
grad_cat_1_train
```

Out[90]:
```
{'Grades_3_5': 0.8549911399881867,
 'Grades_6_8': 0.8388021662949984,
 'Grades_9_12': 0.833249623304872,
 'Grades_PreK_2': 0.8444742620151257}
```

In [0]:
```
grade_neg_train = []
grade_pos_train = []
for i in project_data_train['project_grade_cate
gory']:
    grade_neg_train.append(grad_cat_0_train[i])
    grade_pos_train.append(grad_cat_1_train[i])
```

```
project_data_train['grade_0'] = grade_neg_train
project_data_train['grade_1'] = grade_pos_train
```

In [92]:
```
project_data_train.columns
```

Out[92]:
```
Index(['Unnamed: 0', 'id', 'teacher_id',
'school_state',
       'project_submitted_datetime', 'pro
ject_title',
       'project_resource_summary',
       'teacher_number_of_previously_post
ed_projects', 'clean_categories',
       'clean_subcategories', 'project_gr
ade_category', 'teacher_prefix',
       'essay', 'preprocessed_essays', 'e
ssay_word_count',
       'preprocessed_titles', 'title_word
_count', 'neg', 'neu', 'pos',
       'compound', 'price', 'quantity',
'cat_0', 'cat_1', 'subcat_0',
       'subcat_1', 'state_0', 'state_1',
'prefix_0', 'prefix_1', 'grade_0',
       'grade_1'],
      dtype='object')
```

In [93]:
```
project_data_train.head()
```

Out[93]:

	Unnamed: 0	id	

	Unnamed: 0	id	
0	122998	p144458	911e136a359876dcd
1	133953	p034064	a6e3b4defdd09a10a4
2	116200	p186620	5822478a0332b5f3ba
3	100801	p103335	e787631f3b1451324
4	10536	p204652	66670ae81282034e1

In [94]:
```
cat_0_test
```

Out[94]:
```
{'AppliedLearning': 0.20057306590257878,
 'AppliedLearning Health_Sports': 0.18518
518518518517,
 'AppliedLearning History_Civics': 0.25,
 'AppliedLearning Literacy_Language': 0.1
650485436893204,
 'AppliedLearning Math_Science': 0.166666
66666666666,
 'AppliedLearning Music_Arts': 0.21518987
341772153,
 'AppliedLearning SpecialNeeds': 0.2,
 'AppliedLearning Warmth Care_Hunger': 1.
0,
 'Health_Sports': 0.16228070175438597,
 'Health_Sports AppliedLearning': 0.0,
 'Health_Sports History_Civics': 0.0,
 'Health_Sports Literacy_Language': 0.192
30769230769232,
 'Health_Sports Math_Science': 0.22727272
727272727,
 'Health_Sports Music_Arts': 0.2,
 'Health_Sports SpecialNeeds': 0.19230769
230769232,
 'History_Civics': 0.189873417721519,
 'History_Civics AppliedLearning': 0.5,
 'History_Civics Literacy_Language': 0.05
6,
 'History_Civics Math_Science': 0.1428571
4285714285,
 'History_Civics Music_Arts': 0.137931034
48275862,
 'History_Civics SpecialNeeds': 0.2272727
```

2727272727,
'Literacy_Language': 0.1313501144164759
8,
'Literacy_Language AppliedLearning': 0.2
631578947368421,
'Literacy_Language Health_Sports': 0.142
85714285714285,
'Literacy_Language History_Civics': 0.07
017543859649122,
'Literacy_Language Math_Science': 0.1379
3103448275862,
'Literacy_Language Music_Arts': 0.174698
7951807229,
'Literacy_Language SpecialNeeds': 0.1473
6842105263157,
'Math_Science': 0.1732542819499341,
'Math_Science AppliedLearning': 0.168316
83168316833,
'Math_Science Health_Sports': 0.28571428
57142857,
'Math_Science History_Civics': 0.1568627
450980392,
'Math_Science Literacy_Language': 0.1546
3917525773196,
'Math_Science Music_Arts': 0.14666666666
666667,
'Math_Science SpecialNeeds': 0.188481675
39267016,
'Math_Science Warmth Care_Hunger': 0.333
3333333333333,
'Music_Arts': 0.1040339702760085,
'Music_Arts AppliedLearning': 0.0,
'Music_Arts Health_Sports': 0.0,
'Music_Arts History_Civics': 0.0,
'Music_Arts SpecialNeeds': 0.05263157894
736842,

```
 'SpecialNeeds': 0.19101123595505617,
 'SpecialNeeds Health_Sports': 0.0,
 'SpecialNeeds Music_Arts': 0.25,
 'SpecialNeeds Warmth Care_Hunger': 0.0,
 'Warmth Care_Hunger': 0.0854700854700854
7}
```

In [95]:
```
cat_1_test
```

Out[95]:
```
{'AppliedLearning': 0.7994269340974212,
 'AppliedLearning Health_Sports': 0.81481
48148148148,
 'AppliedLearning History_Civics': 0.75,
 'AppliedLearning Literacy_Language': 0.8
349514563106796,
 'AppliedLearning Math_Science': 0.833333
3333333334,
 'AppliedLearning Music_Arts': 0.78481012
65822784,
 'AppliedLearning SpecialNeeds': 0.8,
 'AppliedLearning Warmth Care_Hunger': 0.
0,
 'Health_Sports': 0.8377192982456141,
 'Health_Sports AppliedLearning': 1.0,
 'Health_Sports History_Civics': 1.0,
 'Health_Sports Literacy_Language': 0.807
6923076923077,
 'Health_Sports Math_Science': 0.77272727
27272727,
 'Health_Sports Music_Arts': 0.8,
 'Health_Sports SpecialNeeds': 0.80769230
76923077,
 'History_Civics': 0.810126582278481,
 'History_Civics AppliedLearning': 0.5,
 'History_Civics Literacy_Language': 0.94
4,
```

'History_Civics Math_Science': 0.8571428
571428571,
'History_Civics Music_Arts': 0.862068965
5172413,
'History_Civics SpecialNeeds': 0.7727272
727272727,
'Literacy_Language': 0.8686498855835241,
'Literacy_Language AppliedLearning': 0.7
368421052631579,
'Literacy_Language Health_Sports': 0.857
1428571428571,
'Literacy_Language History_Civics': 0.92
98245614035088,
'Literacy_Language Math_Science': 0.8620
689655172413,
'Literacy_Language Music_Arts': 0.825301
2048192772,
'Literacy_Language SpecialNeeds': 0.8526
315789473684,
'Math_Science': 0.8267457180500659,
'Math_Science AppliedLearning': 0.831683
1683168316,
'Math_Science Health_Sports': 0.71428571
42857143,
'Math_Science History_Civics': 0.8431372
549019608,
'Math_Science Literacy_Language': 0.8453
60824742268,
'Math_Science Music_Arts': 0.85333333333
33334,
'Math_Science SpecialNeeds': 0.811518324
6073299,
'Math_Science Warmth Care_Hunger': 0.666
6666666666666,
'Music_Arts': 0.8959660297239915,
'Music_Arts AppliedLearning': 1.0,

```
 'Music_Arts Health_Sports': 1.0,
 'Music_Arts History_Civics': 1.0,
 'Music_Arts SpecialNeeds': 0.94736842105
26315,
 'SpecialNeeds': 0.8089887640449438,
 'SpecialNeeds Health_Sports': 1.0,
 'SpecialNeeds Music_Arts': 0.75,
 'SpecialNeeds Warmth Care_Hunger': 1.0,
 'Warmth Care_Hunger': 0.914529914529914
5}
```

In [0]:

```
cat_neg_test = []
cat_pos_test = []
for i in project_data_test['clean_categories']:
    cat_neg_test.append(cat_0_test[i])
    cat_pos_test.append(cat_1_test[i])
project_data_test['cat_0'] = cat_neg_test
project_data_test['cat_1'] = cat_pos_test
```

In [97]:

```
subcat_0_test
```

Out[97]:

```
{'AppliedSciences': 0.15625,
 'AppliedSciences CharacterEducation': 0.
25,
 'AppliedSciences Civics_Government': 0.
0,
 'AppliedSciences College_CareerPrep': 0.
1590909090909091,
 'AppliedSciences CommunityService': 0.0,
 'AppliedSciences ESL': 0.375,
 'AppliedSciences EarlyDevelopment': 0.23
076923076923078,
 'AppliedSciences EnvironmentalScience':
0.18292682926829268,
 'AppliedSciences Extracurricular': 0.214
```

28571428571427,
'AppliedSciences ForeignLanguages': 0.5,
'AppliedSciences Health_LifeScience': 0.
17777777777777778,
'AppliedSciences Health_Wellness': 0.5,
'AppliedSciences History_Geography': 0.2
5,
'AppliedSciences Literacy': 0.1886792452
8301888,
'AppliedSciences Literature_Writing': 0.
10810810810810811,
'AppliedSciences Mathematics': 0.1730769
2307692307,
'AppliedSciences Music': 0.0,
'AppliedSciences Other': 0.1666666666666
6666,
'AppliedSciences ParentInvolvement': 0.1
6666666666666666,
'AppliedSciences PerformingArts': 0.0,
'AppliedSciences SocialSciences': 0.1666
6666666666666,
'AppliedSciences SpecialNeeds': 0.189189
1891891892,
'AppliedSciences VisualArts': 0.11666666
666666667,
'AppliedSciences Warmth Care_Hunger': 0.
0,
'CharacterEducation': 0.3513513513513513
7,
'CharacterEducation Civics_Government':
0.0,
'CharacterEducation College_CareerPrep':
0.1111111111111111,
'CharacterEducation CommunityService':
0.2,
'CharacterEducation ESL': 0.25,

'CharacterEducation EarlyDevelopment': 0.21052631578947367,
'CharacterEducation Extracurricular': 0.25,
'CharacterEducation FinancialLiteracy': 0.0,
'CharacterEducation ForeignLanguages': 0.0,
'CharacterEducation Health_Wellness': 0.125,
'CharacterEducation Literacy': 0.17857142857142858,
'CharacterEducation Literature_Writing': 0.1,
'CharacterEducation Mathematics': 0.3333333333333333,
'CharacterEducation Music': 0.3333333333333333,
'CharacterEducation Other': 0.3333333333333333,
'CharacterEducation ParentInvolvement': 0.0,
'CharacterEducation SocialSciences': 1.0,
'CharacterEducation SpecialNeeds': 0.125,
'CharacterEducation TeamSports': 0.3333333333333333,
'CharacterEducation VisualArts': 0.3333333333333333,
'CharacterEducation Warmth Care_Hunger': 1.0,
'Civics_Government': 0.5714285714285714,
'Civics_Government Economics': 0.0,
'Civics_Government EnvironmentalScience': 0.0,

'Civics_Government Health_LifeScience':
0.0,
'Civics_Government History_Geography':
0.16666666666666666,
'Civics_Government Literacy': 0.11111111
11111111,
'Civics_Government Literature_Writing':
0.08333333333333333,
'Civics_Government Mathematics': 0.5,
'Civics_Government SocialSciences': 0.0,
'Civics_Government SpecialNeeds': 0.6666
666666666666,
'Civics_Government VisualArts': 0.0,
'College_CareerPrep': 0.2162162162162162
3,
'College_CareerPrep CommunityService':
0.0,
'College_CareerPrep ESL': 0.0,
'College_CareerPrep EarlyDevelopment':
0.0,
'College_CareerPrep Economics': 0.0,
'College_CareerPrep EnvironmentalScienc
e': 0.0,
'College_CareerPrep Extracurricular': 0.
16666666666666666,
'College_CareerPrep FinancialLiteracy':
1.0,
'College_CareerPrep ForeignLanguages':
0.0,
'College_CareerPrep Health_LifeScience':
0.0,
'College_CareerPrep Health_Wellness': 0.
5,
'College_CareerPrep Literacy': 0.24,
'College_CareerPrep Literature_Writing':
0.12903225806451613,

'College_CareerPrep Mathematics': 0.0869
5652173913043,
'College_CareerPrep Music': 0.6666666666
666666,
'College_CareerPrep NutritionEducation':
0.0,
'College_CareerPrep Other': 0.1818181818
1818182,
'College_CareerPrep ParentInvolvement':
0.0,
'College_CareerPrep PerformingArts': 0.
0,
'College_CareerPrep SocialSciences': 0.
5,
'College_CareerPrep SpecialNeeds': 0.285
7142857142857,
'College_CareerPrep VisualArts': 0.21428
571428571427,
'CommunityService': 0.2,
'CommunityService Economics': 0.0,
'CommunityService EnvironmentalScience':
0.0,
'CommunityService Extracurricular': 1.0,
'CommunityService Health_LifeScience':
0.0,
'CommunityService Health_Wellness': 0.0,
'CommunityService Literature_Writing':
0.0,
'CommunityService Music': 0.0,
'CommunityService ParentInvolvement': 0.
0,
'CommunityService SpecialNeeds': 0.5,
'CommunityService VisualArts': 0.25,
'ESL': 0.09523809523809523,
'ESL EarlyDevelopment': 0.125,
'ESL EnvironmentalScience': 0.25,

```
 'ESL FinancialLiteracy': 0.0,
 'ESL ForeignLanguages': 0.16666666666666
666,
 'ESL Gym_Fitness': 1.0,
 'ESL Health_LifeScience': 1.0,
 'ESL Health_Wellness': 0.0,
 'ESL History_Geography': 0.0,
 'ESL Literacy': 0.12871287128712872,
 'ESL Literature_Writing': 0.128571428571
42856,
 'ESL Mathematics': 0.2,
 'ESL Music': 0.0,
 'ESL Other': 1.0,
 'ESL ParentInvolvement': 0.0,
 'ESL PerformingArts': 0.0,
 'ESL SocialSciences': 0.0,
 'ESL SpecialNeeds': 0.04347826086956521
6,
 'ESL VisualArts': 0.0,
 'EarlyDevelopment': 0.20454545454545456,
 'EarlyDevelopment EnvironmentalScience':
0.0,
 'EarlyDevelopment ForeignLanguages': 0.
0,
 'EarlyDevelopment Gym_Fitness': 0.666666
6666666666,
 'EarlyDevelopment Health_LifeScience':
0.5,
 'EarlyDevelopment Health_Wellness': 0.11
538461538461539,
 'EarlyDevelopment Literacy': 0.194029850
74626866,
 'EarlyDevelopment Literature_Writing':
0.15,
 'EarlyDevelopment Mathematics': 0.227272
72727272727,
```

'EarlyDevelopment Music': 0.0,
'EarlyDevelopment NutritionEducation':
0.0,
'EarlyDevelopment Other': 0.052631578947
36842,
'EarlyDevelopment ParentInvolvement': 0.
3333333333333333,
'EarlyDevelopment SocialSciences': 0.0,
'EarlyDevelopment SpecialNeeds': 0.17073
170731707318,
'EarlyDevelopment TeamSports': 0.0,
'EarlyDevelopment VisualArts': 0.2857142
857142857,
'Economics': 0.0,
'Economics EnvironmentalScience': 0.0,
'Economics FinancialLiteracy': 0.25,
'Economics History_Geography': 0.0,
'Economics Literacy': 0.0,
'Economics Mathematics': 0.0,
'Economics SocialSciences': 0.0,
'Economics VisualArts': 1.0,
'EnvironmentalScience': 0.15116279069767
44,
'EnvironmentalScience Health_LifeScienc
e': 0.25555555555555554,
'EnvironmentalScience Health_Wellness':
0.2857142857142857,
'EnvironmentalScience History_Geograph
y': 0.08333333333333333,
'EnvironmentalScience Literacy': 0.16129
032258064516,
'EnvironmentalScience Literature_Writin
g': 0.23076923076923078,
'EnvironmentalScience Mathematics': 0.19
56521739130435,
'EnvironmentalScience NutritionEducatio

n': 0.0,
 'EnvironmentalScience ParentInvolvemen
t': 0.0,
 'EnvironmentalScience SocialSciences':
0.0,
 'EnvironmentalScience SpecialNeeds': 0.1
0526315789473684,
 'EnvironmentalScience VisualArts': 0.269
2307692307692,
 'EnvironmentalScience Warmth Care_Hunge
r': 1.0,
 'Extracurricular': 0.1111111111111111,
 'Extracurricular Health_Wellness': 0.333
3333333333333,
 'Extracurricular History_Geography': 0.
0,
 'Extracurricular Literacy': 0.0,
 'Extracurricular Literature_Writing': 0.
0,
 'Extracurricular Mathematics': 0.3333333
333333333,
 'Extracurricular Music': 0.0,
 'Extracurricular Other': 0.0,
 'Extracurricular ParentInvolvement': 0.
0,
 'Extracurricular PerformingArts': 0.0,
 'Extracurricular SpecialNeeds': 0.25,
 'Extracurricular TeamSports': 0.0,
 'Extracurricular VisualArts': 0.30769230
76923077,
 'FinancialLiteracy': 0.1333333333333333
3,
 'FinancialLiteracy History_Geography':
0.0,
 'FinancialLiteracy Literacy': 0.0,
 'FinancialLiteracy Mathematics': 0.11111

11111111111,
 'FinancialLiteracy SpecialNeeds': 0.0,
 'ForeignLanguages': 0.3,
 'ForeignLanguages Gym_Fitness': 0.0,
 'ForeignLanguages History_Geography': 0.
0,
 'ForeignLanguages Literacy': 0.133333333
33333333,
 'ForeignLanguages Literature_Writing':
0.2857142857142857,
 'ForeignLanguages Mathematics': 0.0,
 'ForeignLanguages Music': 0.0,
 'ForeignLanguages SpecialNeeds': 0.0,
 'Gym_Fitness': 0.1650485436893204,
 'Gym_Fitness Health_Wellness': 0.1443298
9690721648,
 'Gym_Fitness Literacy': 0.0,
 'Gym_Fitness Literature_Writing': 0.5,
 'Gym_Fitness Mathematics': 0.0,
 'Gym_Fitness Music': 0.0,
 'Gym_Fitness NutritionEducation': 0.1428
5714285714285,
 'Gym_Fitness Other': 0.0,
 'Gym_Fitness SpecialNeeds': 0.0666666666
6666667,
 'Gym_Fitness TeamSports': 0.239130434782
6087,
 'Health_LifeScience': 0.1884057971014492
8,
 'Health_LifeScience Health_Wellness': 0.
2222222222222222,
 'Health_LifeScience History_Geography':
0.14285714285714285,
 'Health_LifeScience Literacy': 0.0476190
47619047616,
 'Health_LifeScience Literature_Writing':

0.0,
'Health_LifeScience Mathematics': 0.2727272727272727,
'Health_LifeScience NutritionEducation': 0.5,
'Health_LifeScience ParentInvolvement': 0.0,
'Health_LifeScience SocialSciences': 0.2857142857142857,
'Health_LifeScience SpecialNeeds': 0.12,
'Health_LifeScience VisualArts': 0.4,
'Health_LifeScience Warmth Care_Hunger': 0.0,
'Health_Wellness': 0.13864306784660768,
'Health_Wellness History_Geography': 0.0,
'Health_Wellness Literacy': 0.21818181818181817,
'Health_Wellness Literature_Writing': 0.10526315789473684,
'Health_Wellness Mathematics': 0.25,
'Health_Wellness Music': 0.2,
'Health_Wellness NutritionEducation': 0.15151515151515152,
'Health_Wellness Other': 0.0,
'Health_Wellness PerformingArts': 0.0,
'Health_Wellness SpecialNeeds': 0.20535714285714285,
'Health_Wellness TeamSports': 0.1935483870967742,
'Health_Wellness VisualArts': 0.3333333333333333,
'History_Geography': 0.21739130434782608,
'History_Geography Literacy': 0.045454545454545456,

 'History_Geography Literature_Writing':
0.05263157894736842,
 'History_Geography Mathematics': 0.3,
 'History_Geography Music': 0.0,
 'History_Geography Other': 1.0,
 'History_Geography ParentInvolvement':
0.0,
 'History_Geography PerformingArts': 0.0,
 'History_Geography SocialSciences': 0.21
739130434782608,
 'History_Geography SpecialNeeds': 0.1818
181818181818182,
 'History_Geography VisualArts': 0.214285
71428571427,
 'Literacy': 0.10520487264673312,
 'Literacy Literature_Writing': 0.1357552
5812619502,
 'Literacy Mathematics': 0.12817089452603
472,
 'Literacy Music': 0.2,
 'Literacy Other': 0.32,
 'Literacy ParentInvolvement': 0.33333333
33333333,
 'Literacy PerformingArts': 0.25,
 'Literacy SocialSciences': 0.06666666666
666667,
 'Literacy SpecialNeeds': 0.1415929203539
823,
 'Literacy TeamSports': 0.0,
 'Literacy VisualArts': 0.203389830508474
6,
 'Literature_Writing': 0.1757105943152454
7,
 'Literature_Writing Mathematics': 0.1481
4814814814814,
 'Literature_Writing Music': 0.0,

```
 'Literature_Writing Other': 0.2307692307
6923078,
 'Literature_Writing ParentInvolvement':
0.0,
 'Literature_Writing PerformingArts': 0.1
25,
 'Literature_Writing SocialSciences': 0.0
9090909090909091,
 'Literature_Writing SpecialNeeds': 0.179
6875,
 'Literature_Writing TeamSports': 0.0,
 'Literature_Writing VisualArts': 0.16923
076923076924,
 'Mathematics': 0.1518987341772152,
 'Mathematics Music': 0.0,
 'Mathematics Other': 0.0,
 'Mathematics ParentInvolvement': 0.33333
33333333333,
 'Mathematics PerformingArts': 0.5,
 'Mathematics SocialSciences': 0.125,
 'Mathematics SpecialNeeds': 0.2181818181
8181817,
 'Mathematics VisualArts': 0.113636363636
36363,
 'Music': 0.06521739130434782,
 'Music ParentInvolvement': 0.0,
 'Music PerformingArts': 0.11111111111111
11,
 'Music SpecialNeeds': 0.0625,
 'Music VisualArts': 0.0,
 'NutritionEducation': 0.272727272727272
7,
 'NutritionEducation SpecialNeeds': 0.333
3333333333333,
 'NutritionEducation TeamSports': 1.0,
 'Other': 0.19402985074626866,
```

```
 'Other SpecialNeeds': 0.2285714285714285
6,
 'Other VisualArts': 0.0,
 'ParentInvolvement': 0.0,
 'ParentInvolvement SocialSciences': 0.0,
 'ParentInvolvement VisualArts': 0.0,
 'PerformingArts': 0.024390243902439025,
 'PerformingArts SocialSciences': 0.0,
 'PerformingArts SpecialNeeds': 0.0,
 'PerformingArts TeamSports': 0.0,
 'PerformingArts VisualArts': 0.428571428
57142855,
 'SocialSciences': 0.2222222222222222,
 'SocialSciences SpecialNeeds': 0.25,
 'SocialSciences VisualArts': 0.0,
 'SpecialNeeds': 0.19101123595505617,
 'SpecialNeeds TeamSports': 0.0,
 'SpecialNeeds VisualArts': 0.25,
 'SpecialNeeds Warmth Care_Hunger': 0.0,
 'TeamSports': 0.1956521739130435,
 'VisualArts': 0.13541666666666666,
 'Warmth Care_Hunger': 0.0854700854700854
7}
```

In [98]:

```
subcat_1_test
```

Out[98]:

```
{'AppliedSciences': 0.84375,
 'AppliedSciences CharacterEducation': 0.
75,
 'AppliedSciences Civics_Government': 1.
0,
 'AppliedSciences College_CareerPrep': 0.
8409090909090909,
 'AppliedSciences CommunityService': 1.0,
 'AppliedSciences ESL': 0.625,
 'AppliedSciences EarlyDevelopment': 0.76
```

92307692307693,
'AppliedSciences EnvironmentalScience':
0.8170731707317073,
'AppliedSciences Extracurricular': 0.785
7142857142857,
'AppliedSciences ForeignLanguages': 0.5,
'AppliedSciences Health_LifeScience': 0.
8222222222222222,
'AppliedSciences Health_Wellness': 0.5,
'AppliedSciences History_Geography': 0.7
5,
'AppliedSciences Literacy': 0.8113207547
169812,
'AppliedSciences Literature_Writing': 0.
8918918918918919,
'AppliedSciences Mathematics': 0.8269230
769230769,
'AppliedSciences Music': 1.0,
'AppliedSciences Other': 0.8333333333333
334,
'AppliedSciences ParentInvolvement': 0.8
333333333333334,
'AppliedSciences PerformingArts': 1.0,
'AppliedSciences SocialSciences': 0.8333
333333333334,
'AppliedSciences SpecialNeeds': 0.810810
8108108109,
'AppliedSciences VisualArts': 0.88333333
33333333,
'AppliedSciences Warmth Care_Hunger': 1.
0,
'CharacterEducation': 0.648648648648648
7,
'CharacterEducation Civics_Government':
1.0,
'CharacterEducation College_CareerPrep':

0.8888888888888888,
'CharacterEducation CommunityService':
0.8,
'CharacterEducation ESL': 0.75,
'CharacterEducation EarlyDevelopment':
0.7894736842105263,
'CharacterEducation Extracurricular': 0.
75,
'CharacterEducation FinancialLiteracy':
1.0,
'CharacterEducation ForeignLanguages':
1.0,
'CharacterEducation Health_Wellness': 0.
875,
'CharacterEducation Literacy': 0.8214285
714285714,
'CharacterEducation Literature_Writing':
0.9,
'CharacterEducation Mathematics': 0.6666
666666666666,
'CharacterEducation Music': 0.6666666666
666666,
'CharacterEducation Other': 0.6666666666
666666,
'CharacterEducation ParentInvolvement':
1.0,
'CharacterEducation SocialSciences': 0.
0,
'CharacterEducation SpecialNeeds': 0.87
5,
'CharacterEducation TeamSports': 0.66666
66666666666,
'CharacterEducation VisualArts': 0.66666
66666666666,
'CharacterEducation Warmth Care_Hunger':
0.0,

'Civics_Government': 0.4285714285714285
5,
'Civics_Government Economics': 1.0,
'Civics_Government EnvironmentalScienc
e': 1.0,
'Civics_Government Health_LifeScience':
1.0,
'Civics_Government History_Geography':
0.8333333333333334,
'Civics_Government Literacy': 0.88888888
88888888,
'Civics_Government Literature_Writing':
0.9166666666666666,
'Civics_Government Mathematics': 0.5,
'Civics_Government SocialSciences': 1.0,
'Civics_Government SpecialNeeds': 0.3333
333333333333,
'Civics_Government VisualArts': 1.0,
'College_CareerPrep': 0.783783783783783
8,
'College_CareerPrep CommunityService':
1.0,
'College_CareerPrep ESL': 1.0,
'College_CareerPrep EarlyDevelopment':
1.0,
'College_CareerPrep Economics': 1.0,
'College_CareerPrep EnvironmentalScienc
e': 1.0,
'College_CareerPrep Extracurricular': 0.
8333333333333334,
'College_CareerPrep FinancialLiteracy':
0.0,
'College_CareerPrep ForeignLanguages':
1.0,
'College_CareerPrep Health_LifeScience':
1.0,

```
 'College_CareerPrep Health_Wellness': 0.
5,
 'College_CareerPrep Literacy': 0.76,
 'College_CareerPrep Literature_Writing':
0.8709677419354839,
 'College_CareerPrep Mathematics': 0.9130
434782608695,
 'College_CareerPrep Music': 0.3333333333
333333,
 'College_CareerPrep NutritionEducation':
1.0,
 'College_CareerPrep Other': 0.8181818181
818182,
 'College_CareerPrep ParentInvolvement':
1.0,
 'College_CareerPrep PerformingArts': 1.
0,
 'College_CareerPrep SocialSciences': 0.
5,
 'College_CareerPrep SpecialNeeds': 0.714
2857142857143,
 'College_CareerPrep VisualArts': 0.78571
42857142857,
 'CommunityService': 0.8,
 'CommunityService Economics': 1.0,
 'CommunityService EnvironmentalScience':
1.0,
 'CommunityService Extracurricular': 0.0,
 'CommunityService Health_LifeScience':
1.0,
 'CommunityService Health_Wellness': 1.0,
 'CommunityService Literature_Writing':
1.0,
 'CommunityService Music': 1.0,
 'CommunityService ParentInvolvement': 1.
0,
```

```
 'CommunityService SpecialNeeds': 0.5,
 'CommunityService VisualArts': 0.75,
 'ESL': 0.9047619047619048,
 'ESL EarlyDevelopment': 0.875,
 'ESL EnvironmentalScience': 0.75,
 'ESL FinancialLiteracy': 1.0,
 'ESL ForeignLanguages': 0.83333333333333
34,
 'ESL Gym_Fitness': 0.0,
 'ESL Health_LifeScience': 0.0,
 'ESL Health_Wellness': 1.0,
 'ESL History_Geography': 1.0,
 'ESL Literacy': 0.8712871287128713,
 'ESL Literature_Writing': 0.871428571428
5714,
 'ESL Mathematics': 0.8,
 'ESL Music': 1.0,
 'ESL Other': 0.0,
 'ESL ParentInvolvement': 1.0,
 'ESL PerformingArts': 1.0,
 'ESL SocialSciences': 1.0,
 'ESL SpecialNeeds': 0.9565217391304348,
 'ESL VisualArts': 1.0,
 'EarlyDevelopment': 0.7954545454545454,
 'EarlyDevelopment EnvironmentalScience':
1.0,
 'EarlyDevelopment ForeignLanguages': 1.
0,
 'EarlyDevelopment Gym_Fitness': 0.333333
3333333333,
 'EarlyDevelopment Health_LifeScience':
0.5,
 'EarlyDevelopment Health_Wellness': 0.88
46153846153846,
 'EarlyDevelopment Literacy': 0.805970149
2537313,
```

'EarlyDevelopment Literature_Writing': 0.85,
'EarlyDevelopment Mathematics': 0.7727272727272727,
'EarlyDevelopment Music': 1.0,
'EarlyDevelopment NutritionEducation': 1.0,
'EarlyDevelopment Other': 0.9473684210526315,
'EarlyDevelopment ParentInvolvement': 0.6666666666666666,
'EarlyDevelopment SocialSciences': 1.0,
'EarlyDevelopment SpecialNeeds': 0.8292682926829268,
'EarlyDevelopment TeamSports': 1.0,
'EarlyDevelopment VisualArts': 0.7142857142857143,
'Economics': 1.0,
'Economics EnvironmentalScience': 1.0,
'Economics FinancialLiteracy': 0.75,
'Economics History_Geography': 1.0,
'Economics Literacy': 1.0,
'Economics Mathematics': 1.0,
'Economics SocialSciences': 1.0,
'Economics VisualArts': 0.0,
'EnvironmentalScience': 0.8488372093023255,
'EnvironmentalScience Health_LifeScience': 0.7444444444444445,
'EnvironmentalScience Health_Wellness': 0.7142857142857143,
'EnvironmentalScience History_Geography': 0.9166666666666666,
'EnvironmentalScience Literacy': 0.8387096774193549,
'EnvironmentalScience Literature_Writin

g': 0.7692307692307693,
 'EnvironmentalScience Mathematics': 0.8043478260869565,
 'EnvironmentalScience NutritionEducation': 1.0,
 'EnvironmentalScience ParentInvolvement': 1.0,
 'EnvironmentalScience SocialSciences': 1.0,
 'EnvironmentalScience SpecialNeeds': 0.8947368421052632,
 'EnvironmentalScience VisualArts': 0.7307692307692307,
 'EnvironmentalScience Warmth Care_Hunger': 0.0,
 'Extracurricular': 0.8888888888888888,
 'Extracurricular Health_Wellness': 0.6666666666666666,
 'Extracurricular History_Geography': 1.0,
 'Extracurricular Literacy': 1.0,
 'Extracurricular Literature_Writing': 1.0,
 'Extracurricular Mathematics': 0.6666666666666666,
 'Extracurricular Music': 1.0,
 'Extracurricular Other': 1.0,
 'Extracurricular ParentInvolvement': 1.0,
 'Extracurricular PerformingArts': 1.0,
 'Extracurricular SpecialNeeds': 0.75,
 'Extracurricular TeamSports': 1.0,
 'Extracurricular VisualArts': 0.6923076923076923,
 'FinancialLiteracy': 0.8666666666666667,
 'FinancialLiteracy History_Geography':

1.0,
'FinancialLiteracy Literacy': 1.0,
'FinancialLiteracy Mathematics': 0.88888
88888888888,
'FinancialLiteracy SpecialNeeds': 1.0,
'ForeignLanguages': 0.7,
'ForeignLanguages Gym_Fitness': 1.0,
'ForeignLanguages History_Geography': 1.
0,
'ForeignLanguages Literacy': 0.866666666
6666667,
'ForeignLanguages Literature_Writing':
0.7142857142857143,
'ForeignLanguages Mathematics': 1.0,
'ForeignLanguages Music': 1.0,
'ForeignLanguages SpecialNeeds': 1.0,
'Gym_Fitness': 0.8349514563106796,
'Gym_Fitness Health_Wellness': 0.8556701
030927835,
'Gym_Fitness Literacy': 1.0,
'Gym_Fitness Literature_Writing': 0.5,
'Gym_Fitness Mathematics': 1.0,
'Gym_Fitness Music': 1.0,
'Gym_Fitness NutritionEducation': 0.8571
428571428571,
'Gym_Fitness Other': 1.0,
'Gym_Fitness SpecialNeeds': 0.9333333333
333333,
'Gym_Fitness TeamSports': 0.760869565217
3914,
'Health_LifeScience': 0.811594202898550
8,
'Health_LifeScience Health_Wellness': 0.
7777777777777778,
'Health_LifeScience History_Geography':
0.8571428571428571,

'Health_LifeScience Literacy': 0.9523809
523809523,
'Health_LifeScience Literature_Writing':
1.0,
'Health_LifeScience Mathematics': 0.7272
727272727273,
'Health_LifeScience NutritionEducation':
0.5,
'Health_LifeScience ParentInvolvement':
1.0,
'Health_LifeScience SocialSciences': 0.7
142857142857143,
'Health_LifeScience SpecialNeeds': 0.88,
'Health_LifeScience VisualArts': 0.6,
'Health_LifeScience Warmth Care_Hunger':
1.0,
'Health_Wellness': 0.8613569321533924,
'Health_Wellness History_Geography': 1.
0,
'Health_Wellness Literacy': 0.7818181818
181819,
'Health_Wellness Literature_Writing': 0.
8947368421052632,
'Health_Wellness Mathematics': 0.75,
'Health_Wellness Music': 0.8,
'Health_Wellness NutritionEducation': 0.
8484848484848485,
'Health_Wellness Other': 1.0,
'Health_Wellness PerformingArts': 1.0,
'Health_Wellness SpecialNeeds': 0.794642
8571428571,
'Health_Wellness TeamSports': 0.80645161
29032258,
'Health_Wellness VisualArts': 0.66666666
66666666,
'History_Geography': 0.782608695652174,

'History_Geography Literacy': 0.95454545
45454546,
'History_Geography Literature_Writing':
0.9473684210526315,
'History_Geography Mathematics': 0.7,
'History_Geography Music': 1.0,
'History_Geography Other': 0.0,
'History_Geography ParentInvolvement':
1.0,
'History_Geography PerformingArts': 1.0,
'History_Geography SocialSciences': 0.78
2608695652174,
'History_Geography SpecialNeeds': 0.8181
818181818182,
'History_Geography VisualArts': 0.785714
2857142857,
'Literacy': 0.8947951273532669,
'Literacy Literature_Writing': 0.8642447
41873805,
'Literacy Mathematics': 0.87182910547396
53,
'Literacy Music': 0.8,
'Literacy Other': 0.68,
'Literacy ParentInvolvement': 0.66666666
66666666,
'Literacy PerformingArts': 0.75,
'Literacy SocialSciences': 0.93333333333
33333,
'Literacy SpecialNeeds': 0.8584070796460
177,
'Literacy TeamSports': 1.0,
'Literacy VisualArts': 0.796610169491525
4,
'Literature_Writing': 0.824289405684754
5,
'Literature_Writing Mathematics': 0.8518

518518518519,
 'Literature_Writing Music': 1.0,
 'Literature_Writing Other': 0.7692307692
307693,
 'Literature_Writing ParentInvolvement':
1.0,
 'Literature_Writing PerformingArts': 0.8
75,
 'Literature_Writing SocialSciences': 0.9
090909090909091,
 'Literature_Writing SpecialNeeds': 0.820
3125,
 'Literature_Writing TeamSports': 1.0,
 'Literature_Writing VisualArts': 0.83076
92307692308,
 'Mathematics': 0.8481012658227848,
 'Mathematics Music': 1.0,
 'Mathematics Other': 1.0,
 'Mathematics ParentInvolvement': 0.66666
66666666666,
 'Mathematics PerformingArts': 0.5,
 'Mathematics SocialSciences': 0.875,
 'Mathematics SpecialNeeds': 0.7818181818
181819,
 'Mathematics VisualArts': 0.886363636363
6364,
 'Music': 0.9347826086956522,
 'Music ParentInvolvement': 1.0,
 'Music PerformingArts': 0.88888888888888
88,
 'Music SpecialNeeds': 0.9375,
 'Music VisualArts': 1.0,
 'NutritionEducation': 0.727272727272727
3,
 'NutritionEducation SpecialNeeds': 0.666
6666666666666,

```
 'NutritionEducation TeamSports': 0.0,
 'Other': 0.8059701492537313,
 'Other SpecialNeeds': 0.771428571428571
5,
 'Other VisualArts': 1.0,
 'ParentInvolvement': 1.0,
 'ParentInvolvement SocialSciences': 1.0,
 'ParentInvolvement VisualArts': 1.0,
 'PerformingArts': 0.975609756097561,
 'PerformingArts SocialSciences': 1.0,
 'PerformingArts SpecialNeeds': 1.0,
 'PerformingArts TeamSports': 1.0,
 'PerformingArts VisualArts': 0.571428571
4285714,
 'SocialSciences': 0.7777777777777778,
 'SocialSciences SpecialNeeds': 0.75,
 'SocialSciences VisualArts': 1.0,
 'SpecialNeeds': 0.8089887640449438,
 'SpecialNeeds TeamSports': 1.0,
 'SpecialNeeds VisualArts': 0.75,
 'SpecialNeeds Warmth Care_Hunger': 1.0,
 'TeamSports': 0.8043478260869565,
 'VisualArts': 0.8645833333333334,
 'Warmth Care_Hunger': 0.914529914529914
5}
```

In [0]:

```
subcat_neg_test = []
subcat_pos_test = []
for i in project_data_test['clean_subcategorie
s']:
    subcat_neg_test.append(subcat_0_test[i])
    subcat_pos_test.append(subcat_1_test[i])
project_data_test['subcat_0'] = subcat_neg_test
project_data_test['subcat_1'] = subcat_pos_test
```

In [100]:
```
state_0_test
```

Out[100]:
```
{'AK': 0.16666666666666666,
 'AL': 0.1497005988023952,
 'AR': 0.1744186046511628,
 'AZ': 0.14347826086956522,
 'CA': 0.14559386973180077,
 'CO': 0.13559322033898305,
 'CT': 0.14965986394557823,
 'DC': 0.18867924528301888,
 'DE': 0.11764705882352941,
 'FL': 0.17391304347826086,
 'GA': 0.17204301075268819,
 'HI': 0.18518518518518517,
 'IA': 0.09433962264150944,
 'ID': 0.20833333333333334,
 'IL': 0.1597051597051597,
 'IN': 0.17054263565891473,
 'KS': 0.12121212121212122,
 'KY': 0.13761467889908258,
 'LA': 0.13615023474178403,
 'MA': 0.13924050632911392,
 'MD': 0.15037593984962405,
 'ME': 0.15384615384615385,
 'MI': 0.15037593984962405,
 'MN': 0.136,
 'MO': 0.16055045871559634,
 'MS': 0.15315315315315314,
 'MT': 0.2,
 'NC': 0.16488222698072805,
 'ND': 0.125,
 'NE': 0.24324324324324326,
 'NH': 0.06666666666666667,
 'NJ': 0.18604651162790697,
 'NM': 0.09433962264150944,
 'NV': 0.171875,
```

```
 'NY': 0.15030674846625766,
 'OH': 0.13524590163934427,
 'OK': 0.14691943127962084,
 'OR': 0.1893939393939394,
 'PA': 0.14634146341463414,
 'RI': 0.13043478260869565,
 'SC': 0.15300546448087432,
 'SD': 0.2727272727272727,
 'TN': 0.15204678362573099,
 'TX': 0.17298937784522003,
 'UT': 0.1518987341772152,
 'VA': 0.10059171597633136,
 'VT': 1.0,
 'WA': 0.09345794392523364,
 'WI': 0.1695906432748538,
 'WV': 0.15217391304347827,
 'WY': 0.5}
```

In [101]:
```
state_1_test
```

Out[101]:
```
{'AK': 0.8333333333333334,
 'AL': 0.8502994011976048,
 'AR': 0.8255813953488372,
 'AZ': 0.8565217391304348,
 'CA': 0.8544061302681992,
 'CO': 0.864406779661017,
 'CT': 0.8503401360544217,
 'DC': 0.8113207547169812,
 'DE': 0.8823529411764706,
 'FL': 0.8260869565217391,
 'GA': 0.8279569892473119,
 'HI': 0.8148148148148148,
 'IA': 0.9056603773584906,
 'ID': 0.7916666666666666,
 'IL': 0.8402948402948403,
 'IN': 0.8294573643410853,
```

```
'KS': 0.8787878787878788,
'KY': 0.8623853211009175,
'LA': 0.863849765258216,
'MA': 0.8607594936708861,
'MD': 0.849624060150376,
'ME': 0.8461538461538461,
'MI': 0.849624060150376,
'MN': 0.864,
'MO': 0.8394495412844036,
'MS': 0.8468468468468469,
'MT': 0.8,
'NC': 0.8351177730192719,
'ND': 0.875,
'NE': 0.7567567567567568,
'NH': 0.9333333333333333,
'NJ': 0.813953488372093,
'NM': 0.9056603773584906,
'NV': 0.828125,
'NY': 0.8496932515337423,
'OH': 0.8647540983606558,
'OK': 0.8530805687203792,
'OR': 0.8106060606060606,
'PA': 0.8536585365853658,
'RI': 0.8695652173913043,
'SC': 0.8469945355191257,
'SD': 0.7272727272727273,
'TN': 0.847953216374269,
'TX': 0.8270106221547799,
'UT': 0.8481012658227848,
'VA': 0.8994082840236687,
'VT': 0.0,
'WA': 0.9065420560747663,
'WI': 0.8304093567251462,
'WV': 0.8478260869565217,
'WY': 0.5}
```

In [0]:
```
state_neg_test = []
state_pos_test = []
for i in project_data_test['school_state']:
    state_neg_test.append(state_0_test[i])
    state_pos_test.append(state_1_test[i])
project_data_test['state_0'] = state_neg_test
project_data_test['state_1'] = state_pos_test
```

In [103]:
```
prefix_0_test
```

Out[103]:
```
{'Mr': 0.13846153846153847,
 'Mrs': 0.15196926032660904,
 'Ms': 0.15955056179775282,
 'Teacher': 0.17777777777777778}
```

In [104]:
```
prefix_1_test
```

Out[104]:
```
{'Mr': 0.8615384615384616,
 'Mrs': 0.8480307396733909,
 'Ms': 0.8404494382022472,
 'Teacher': 0.8222222222222222}
```

In [0]:
```
prefix_neg_test = []
prefix_pos_test = []
for i in project_data_test['teacher_prefix']:
    prefix_neg_test.append(prefix_0_test[i])
    prefix_pos_test.append(prefix_1_test[i])
project_data_test['prefix_0'] = prefix_neg_test
project_data_test['prefix_1'] = prefix_pos_test
```

In [106]:
```
grad_cat_0_test
```

Out[106]:
```
{'Grades_3_5': 0.15200708382526565,
 'Grades_6_8': 0.17060367454068243,
```

```
 'Grades_9_12': 0.14765784114052954,
 'Grades_PreK_2': 0.1510234648027958}
```

In [107]:
```
grad_cat_1_test
```

Out[107]:
```
{'Grades_3_5': 0.8479929161747344,
 'Grades_6_8': 0.829396325459317б,
 'Grades_9_12': 0.8523421588594705,
 'Grades_PreK_2': 0.8489765351972042}
```

In [0]:
```
grade_neg_test = []
grade_pos_test = []
for i in project_data_test['project_grade_categ
ory']:
    grade_neg_test.append(grad_cat_0_test[i])
    grade_pos_test.append(grad_cat_1_test[i])
project_data_test['grade_0'] = grade_neg_test
project_data_test['grade_1'] = grade_pos_test
```

In [109]:
```
project_data_test.columns
```

Out[109]:
```
Index(['Unnamed: 0', 'id', 'teacher_id',
'school_state',
       'project_submitted_datetime', 'pro
ject_title',
       'project_resource_summary',
       'teacher_number_of_previously_post
ed_projects', 'clean_categories',
       'clean_subcategories', 'project_gr
ade_category', 'teacher_prefix',
       'essay', 'preprocessed_essays', 'e
ssay_word_count',
       'preprocessed_titles', 'title_word
_count', 'neg', 'neu', 'pos',
       'compound', 'price', 'quantity',
```

```
'cat_0', 'cat_1', 'subcat_0',
       'subcat_1', 'state_0', 'state_1',
'prefix_0', 'prefix_1', 'grade_0',
       'grade_1'],
      dtype='object')
```

In [110]:

```
project_data_test.head()
```

Out[110]:

	Unnamed: 0	id	
0	121419	p142157	e2d3b7f6d24e6ee02(
1	155129	p157153	024f8b5c9a3cb36e1a
2	165704	p046037	34b27c2f81c204041(
3	83501	p017261	cdcfe0ff6ec65309974

	Unnamed: 0	id	
4	39939	p030597	1b2049519ca43f594(

In [111]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, go
t 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53
709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a singl
e feature
# array.reshape(1, -1)  if it contains a single
sample.

normalizer.fit(project_data_train["cat_0"].valu
es.reshape(-1,1))  #fit has to be done only on
 Train data

cat_0_train_normalized = normalizer.transform(p
roject_data_train["cat_0"].values.reshape(1,-1
))
cat_0_test_normalized = normalizer.transform(pr
oject_data_test["cat_0"].values.reshape(1,-1))
```

```python
#reshaping after normalizing
cat_0_train_normalized = cat_0_train_normalized
.reshape(-1,1)
cat_0_test_normalized = cat_0_test_normalized.r
eshape(-1,1)

print("After vectorizations")
print(cat_0_train_normalized.shape, y_train.sha
pe)
print(cat_0_test_normalized.shape, y_test.shape
)
```

```
After vectorizations
(20100, 1) (20100,)
(9900, 1) (9900,)
```

In [112]:

```python
cat_0_train_normalized
```

Out[112]:

```
array([[0.00599631],
       [0.00811779],
       [0.00923005],
       ...,
       [0.00811779],
       [0.00811779],
       [0.00598429]])
```

In [113]:

```python
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, go
t 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53
709.67].
```

```
# Reshape your data either using
# array.reshape(-1, 1) if your data has a singl
e feature
# array.reshape(1, -1)  if it contains a single
sample.

normalizer.fit(project_data_train["cat_1"].valu
es.reshape(-1,1))   #fit has to be done only on
Train data

cat_1_train_normalized = normalizer.transform(p
roject_data_train["cat_1"].values.reshape(1,-1
))
cat_1_test_normalized = normalizer.transform(pr
oject_data_test["cat_1"].values.reshape(1,-1))


#reshaping after normalizing
cat_1_train_normalized = cat_1_train_normalized
.reshape(-1,1)
cat_1_test_normalized = cat_1_test_normalized.r
eshape(-1,1)

print("After vectorizations")
print(cat_1_train_normalized.shape, y_train.sha
pe)
print(cat_1_test_normalized.shape, y_test.shape
)
```

```
After vectorizations
(20100, 1) (20100,)
(9900, 1) (9900,)
```

In [114]:

```
from sklearn.preprocessing import Normalizer
```

```
normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, go
t 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53
709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a singl
e feature
# array.reshape(1, -1)  if it contains a single
sample.

normalizer.fit(project_data_train["subcat_0"].v
alues.reshape(-1,1))  #fit has to be done only
 on Train data

subcat_0_train_normalized = normalizer.transfor
m(project_data_train["subcat_0"].values.reshape
(1,-1))
subcat_0_test_normalized = normalizer.transform
(project_data_test["subcat_0"].values.reshape(1
,-1))

#reshaping after normalizing
subcat_0_train_normalized = subcat_0_train_norm
alized.reshape(-1,1)
subcat_0_test_normalized = subcat_0_test_normal
ized.reshape(-1,1)




print("After vectorizations")
print(subcat_0_train_normalized.shape, y_train.
shape)
```

```
print(subcat_0_test_normalized.shape, y_test.shape)
```

```
After vectorizations
(20100, 1) (20100,)
(9900, 1) (9900,)
```

In [115]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

normalizer.fit(project_data_train["subcat_1"].values.reshape(-1,1))  #fit has to be done only on Train data

subcat_1_train_normalized = normalizer.transform(project_data_train["subcat_1"].values.reshape(1,-1))
subcat_1_test_normalized = normalizer.transform(project_data_test["subcat_1"].values.reshape(1,-1))



#reshaping after normalizing
```

```
subcat_1_train_normalized = subcat_1_train_norm
alized.reshape(-1,1)
subcat_1_test_normalized = subcat_1_test_normal
ized.reshape(-1,1)

print("After vectorizations")
print(subcat_1_train_normalized.shape, y_train.
shape)
print(subcat_1_test_normalized.shape, y_test.sh
ape)
```

```
After vectorizations
(20100, 1) (20100,)
(9900, 1) (9900,)
```

In [116]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, go
t 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53
709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a singl
e feature
# array.reshape(1, -1)  if it contains a single
sample.

normalizer.fit(project_data_train["state_0"].va
lues.reshape(-1,1))  #fit has to be done only o
n Train data

state_0_train_normalized = normalizer.transform
```

```
(project_data_train["state_0"].values.reshape(1
,-1))
state_0_test_normalized = normalizer.transform(
project_data_test["state_0"].values.reshape(1,-
1))

#reshaping after normalizing
state_0_train_normalized = state_0_train_normal
ized.reshape(-1,1)
state_0_test_normalized = state_0_test_normaliz
ed.reshape(-1,1)

print("After vectorizations")
print(state_0_train_normalized.shape, y_train.s
hape)
print(state_0_test_normalized.shape, y_test.sha
pe)
```

```
After vectorizations
(20100, 1) (20100,)
(9900, 1) (9900,)
```

In [117]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, go
t 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53
709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a singl
e feature
# array.reshape(1, -1)  if it contains a single
```

```
sample.

normalizer.fit(project_data_train["state_1"].values.reshape(-1,1))  #fit has to be done only on Train data

state_1_train_normalized = normalizer.transform(project_data_train["state_1"].values.reshape(1,-1))
state_1_test_normalized = normalizer.transform(project_data_test["state_1"].values.reshape(1,-1))



#reshaping after normalizing
state_1_train_normalized = state_1_train_normalized.reshape(-1,1)
state_1_test_normalized = state_1_test_normalized.reshape(-1,1)

print("After vectorizations")
print(state_1_train_normalized.shape, y_train.shape)
print(state_1_test_normalized.shape, y_test.shape)
```

```
After vectorizations
(20100, 1) (20100,)
(9900, 1) (9900,)
```

In [118]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()
```

```
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, go
t 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53
709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a singl
e feature
# array.reshape(1, -1)  if it contains a single
sample.

normalizer.fit(project_data_train["prefix_0"].v
alues.reshape(-1,1))  #fit has to be done only
 on Train data

prefix_0_train_normalized = normalizer.transfor
m(project_data_train["prefix_0"].values.reshape
(1,-1))
prefix_0_test_normalized = normalizer.transform
(project_data_test["prefix_0"].values.reshape(1
,-1))

#reshaping after normalizing
prefix_0_train_normalized = prefix_0_train_norm
alized.reshape(-1,1)
prefix_0_test_normalized = prefix_0_test_normal
ized.reshape(-1,1)

print("After vectorizations")
print(prefix_0_train_normalized.shape, y_train.
shape)
print(prefix_0_test_normalized.shape, y_test.sh
ape)
```

```
After vectorizations
(20100, 1) (20100,)
```

(9900, 1) (9900,)

In [119]:
```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, go
t 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53
709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a singl
e feature
# array.reshape(1, -1)  if it contains a single
sample.

normalizer.fit(project_data_train["prefix_1"].v
alues.reshape(-1,1))  #fit has to be done only
 on Train data

prefix_1_train_normalized = normalizer.transfor
m(project_data_train["prefix_1"].values.reshape
(1,-1))
prefix_1_test_normalized = normalizer.transform
(project_data_test["prefix_1"].values.reshape(1
,-1))


#reshaping after normalizing
prefix_1_train_normalized = prefix_1_train_norm
alized.reshape(-1,1)
prefix_1_test_normalized = prefix_1_test_normal
ized.reshape(-1,1)
```

```
print("After vectorizations")
print(prefix_1_train_normalized.shape, y_train.shape)
print(prefix_1_test_normalized.shape, y_test.shape)
```

```
After vectorizations
(20100, 1) (20100,)
(9900, 1) (9900,)
```

In [120]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

normalizer.fit(project_data_train["grade_0"].values.reshape(-1,1))  #fit has to be done only on Train data

grade_0_train_normalized = normalizer.transform(project_data_train["grade_0"].values.reshape(1,-1))
grade_0_test_normalized = normalizer.transform(project_data_test["grade_0"].values.reshape(1,-
```

```
1))

#reshaping after normalizing
grade_0_train_normalized = grade_0_train_normal
ized.reshape(-1,1)
grade_0_test_normalized = grade_0_test_normaliz
ed.reshape(-1,1)




print("After vectorizations")
print(grade_0_train_normalized.shape, y_train.s
hape)
print(grade_0_test_normalized.shape, y_test.sha
pe)
```

```
After vectorizations
(20100, 1) (20100,)
(9900, 1) (9900,)
```

In [121]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, go
t 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53
709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a singl
e feature
# array.reshape(1, -1)  if it contains a single
sample.
```

```
normalizer.fit(project_data_train["grade_1"].values.reshape(-1,1))  #fit has to be done only on Train data

grade_1_train_normalized = normalizer.transform(project_data_train["grade_1"].values.reshape(1,-1))
grade_1_test_normalized = normalizer.transform(project_data_test["grade_1"].values.reshape(1,-1))



#reshaping after normalizing
grade_1_train_normalized = grade_1_train_normalized.reshape(-1,1)
grade_1_test_normalized = grade_1_test_normalized.reshape(-1,1)

print("After vectorizations")
print(grade_1_train_normalized.shape, y_train.shape)
print(grade_1_test_normalized.shape, y_test.shape)
```

```
After vectorizations
(20100, 1) (20100,)
(9900, 1) (9900,)
```

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

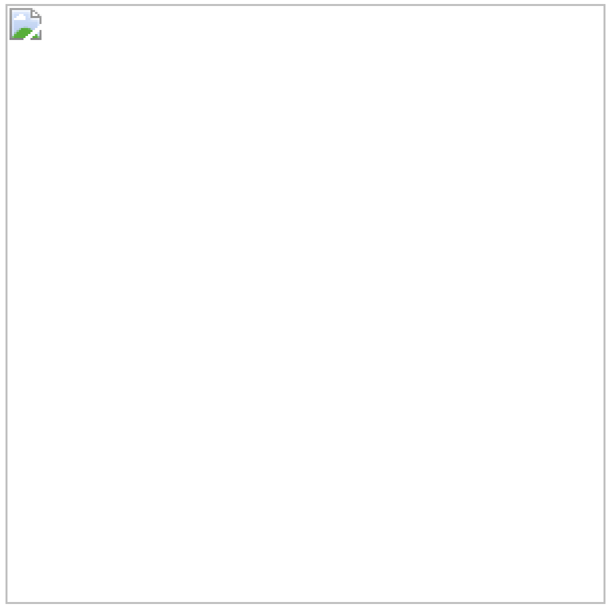
3. **Representation of results**

- 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 explains how to plot this 3d plot, you can find it in the same drive *3d_scatter_plot.ipynb*

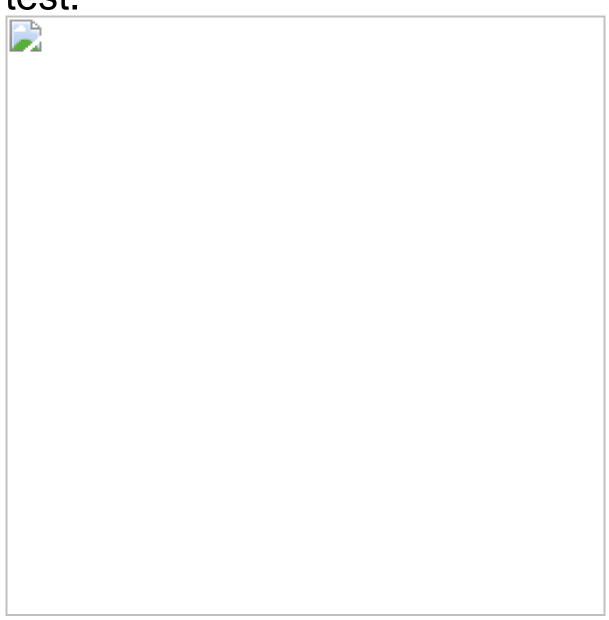
or

- 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-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this [link.]

2. Random Forest and

GBDT

2.4 Applying Random Forest

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

SET 1: categorical(instead of one hot encoding, try response coding(using probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)

In [0]:

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

X_train = hstack((cat_0_train_normalized, cat_1_train_normalized, subcat_0_train_normalized, subcat_1_train_normalized, state_0_train_normalized, state_1_train_normalized, grade_0_train_normalized, grade_1_train_normalized, prefix_0_train_normalized, prefix_1_train_normalized, price_normalized_train, quantity_normalized_train, previously_posted_projects_normalized_train, title_word_count_normalized_train, essay_word_count_normalized_train, sent_pos_train, sent_neg_train, sent_neu_train, sent_compound_train, train_title_bow, train_essay_bow)).tocsr()
```

```
X_test =  hstack((cat_0_test_normalized, cat_1_test_normalized, subcat_0_test_normalized, subcat_1_test_normalized, state_0_test_normalized, state_1_test_normalized, grade_0_test_normalized, grade_1_test_normalized, prefix_0_test_normalized, prefix_1_test_normalized, price_normalized_test, quantity_normalized_test, previously_posted_projects_normalized_test, title_word_count_normalized_test, essay_word_count_normalized_test, sent_pos_test, sent_neg_test, sent_neu_test, sent_compound_test, test_title_bow, test_essay_bow)).tocsr()
```

In [129]:

```
print(X_train.shape)
print(X_test.shape)
```

```
(20100, 9516)
(9900, 9516)
```

In [133]:

```
%%time
# https://medium.com/@erikgreenj/k-neighbors-classifier-with-gridsearchcv-basics-3c445ddeb657

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

rf = RandomForestClassifier(class_weight='balanced')

grid_params = {'n_estimators': [10, 50, 100, 150, 200, 300, 500, 1000], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 10]}
```

```
gs = GridSearchCV(rf, grid_params, cv=3, scorin
g='roc_auc',return_train_score = True, n_jobs =
-1)
gs_results = gs.fit(X_train, y_train)
print(gs_results.best_score_)
print(gs_results.best_estimator_)
print(gs_results.best_params_)
```

```
0.7131973322693029
RandomForestClassifier(bootstrap=True, cl
ass_weight='balanced',
                       criterion='gini',
max_depth=9, max_features='auto',
                       max_leaf_nodes=Non
e, min_impurity_decrease=0.0,
                       min_impurity_split
=None, min_samples_leaf=1,
                       min_samples_split=
2, min_weight_fraction_leaf=0.0,
                       n_estimators=1000,
n_jobs=None, oob_score=False,
                       random_state=None,
verbose=0, warm_start=False)
{'max_depth': 9, 'n_estimators': 1000}
CPU times: user 15 s, sys: 122 ms, total:
15.1 s
Wall time: 9min 44s
```

In [134]:

```
#Output of GridSearchCV
print('Best score: ',gs_results.best_score_)
print('k value with best score: ',gs_results.be
st_params_)
print('='*75)
print('Train AUC scores')
print(gs.cv_results_['mean_train_score'])
```

```python
print('CV AUC scores')
print(gs.cv_results_['mean_test_score'])
```

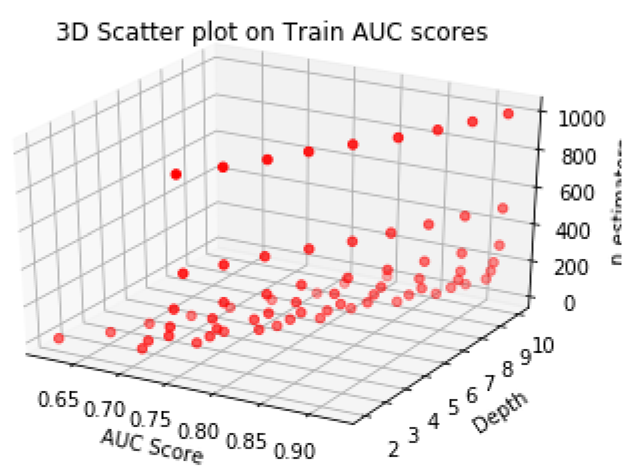
```
Best score:  0.7131973322693029
k value with best score:  {'max_depth':
9, 'n_estimators': 1000}
=========================================
==================================
Train AUC scores
[0.6197024  0.71118673 0.7180743  0.74052
739 0.74193424 0.74644417
 0.75729199 0.75254178 0.65266872 0.74678
729 0.7608732  0.76886081
 0.76401181 0.77629182 0.77686949 0.77760
727 0.67172212 0.75312819
 0.79098932 0.785831   0.80120947 0.79947
703 0.79750177 0.80057116
 0.69482393 0.78824295 0.80217959 0.81390
851 0.80854355 0.81404899
 0.82188174 0.82089658 0.71473741 0.81284
206 0.82049571 0.83458714
 0.8277752  0.84148761 0.84572109 0.84458
971 0.73844204 0.82489382
 0.84232182 0.84886012 0.856437   0.86276
143 0.86533837 0.87008174
 0.76672316 0.85363002 0.86614668 0.87934
331 0.87659776 0.88136262
 0.88435987 0.89007441 0.77578522 0.87520
258 0.89096766 0.89811094
 0.90041549 0.900678   0.90245933 0.90573
581 0.8019737  0.88724952
 0.90878597 0.91323887 0.91583302 0.92103
13  0.92249553 0.92294099]
CV AUC scores
[0.58882079 0.65866565 0.66984857 0.67835
435 0.68249781 0.68993669
```

```
 0.69713029 0.69199775 0.60335395 0.67211
655 0.68337308 0.69318119
 0.69093876 0.69511778 0.69671805 0.70009
407 0.60788035 0.66672327
 0.69408824 0.69224479 0.70255156 0.70246
372 0.69979355 0.70270253
 0.61404366 0.68095333 0.68531186 0.70425
13  0.69629887 0.70227142
 0.70532589 0.70322977 0.61859542 0.68869
41  0.6953417  0.70145882
 0.69692243 0.70452543 0.70698895 0.70630
441 0.62716049 0.67795005
 0.69954042 0.69561302 0.70279206 0.70596
657 0.70725859 0.71194647
 0.64340328 0.68395127 0.69667027 0.70187
992 0.7031952  0.70622006
 0.70844573 0.71048152 0.63717661 0.68484
582 0.69623294 0.70668566
 0.70647437 0.70808881 0.70834151 0.71319
733 0.6454999  0.68756015
 0.69631357 0.70488733 0.70769435 0.70747
275 0.7094051  0.71183441]
```

In [135]:

```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt



fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

g1 = list(gs.cv_results_['mean_train_score'])
#Train AUC Score
g2 = [2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,4,4,4,4,
4,4,4,4,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,7,7,7,
```

```
7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,10,1
0,10,10,10,10,10,10] #Depth
g3 = [10, 50, 100, 150, 200, 300, 500, 1000,10,
50, 100, 150, 200, 300, 500, 1000,10, 50, 100,
150, 200, 300, 500, 1000,10, 50, 100, 150, 200,
300, 500, 1000,10, 50, 100, 150, 200, 300, 500,
1000,10, 50, 100, 150, 200, 300, 500, 1000,10,
50, 100, 150, 200, 300, 500, 1000,10, 50, 100,
150, 200, 300, 500, 1000,10, 50, 100, 150, 200,
300, 500, 1000] #n_estimators



ax.scatter(g1, g2, g3, c='r', marker='o')

ax.set_xlabel('AUC Score')
ax.set_ylabel('Depth')
ax.set_zlabel('n_estimators')

plt.title('3D Scatter plot on Train AUC scores'
)
plt.show()
```



In [136]:
```
gs.cv_results_
```
Out[136]:
```
{'mean_fit_time': array([ 0.10730672,  0.
27795307,  0.4920752 ,  0.70988552,  0.94
027162,
         1.37299554,  2.22590407,  4.3812
8575,  0.10141945,  0.31458227,
         0.56633878,  0.82869252,  1.0878
5629,  1.59666451,  2.66094605,
         5.25266202,  0.11657818,  0.3599
9942,  0.68160033,  0.95723494,
         1.27223523,  1.89552673,  3.1094
7053,  6.32561278,  0.13127065,
         0.4297742 ,  0.79475737,  1.1792
4476,  1.59764004,  2.25806729,
         3.81356454,  7.44476151,  0.1353
058 ,  0.49965668,  0.94490242,
         1.37959313,  1.8296059 ,  2.6840
121 ,  4.41460347,  9.06667304,
         0.16016078,  0.58060225,  1.0904
0642,  1.67294518,  2.20483851,
         3.11990356,  5.16865015, 10.2535
1111,  0.17527731,  0.66968489,
         1.23025513,  1.86094189,  2.3874
2908,  3.62667084,  5.97094798,
        11.84175976,  0.1915435 ,  0.7564
342 ,  1.40529331,  2.15697153,
         2.81129821,  4.18105618,  6.9099
3555, 13.89713216,  0.21022503,
         0.84958553,  1.65975849,  2.4834
8951,  3.28290288,  4.86647964,
         8.03348859, 15.06032546]),
 'mean_score_time': array([0.03968461, 0.
15108267, 0.30749997, 0.47255365, 0.61811
177,
        0.87932483, 1.45650045, 2.9486518
```

```
7, 0.03589217, 0.1721646 ,
        0.34348353, 0.47362328, 0.5729801
7, 0.90592146, 1.43371725,
        2.97248785, 0.0346752 , 0.1752584
8, 0.34932812, 0.43509841,
        0.58143981, 0.87473273, 1.4372763
6, 2.97915467, 0.0350066 ,
        0.16767613, 0.34275786, 0.4752948
3, 0.59764314, 0.85493755,
        1.47883463, 2.97243404, 0.0354956
8, 0.16911356, 0.34514753,
        0.43461808, 0.54898087, 0.8684261
6, 1.39629539, 3.03963868,
        0.0350825 , 0.14636628, 0.3449156
3, 0.50586057, 0.63385161,
        0.91623537, 1.47806168, 2.8996207
7, 0.03619766, 0.16174658,
        0.36349773, 0.45146028, 0.6112378
4, 0.89037991, 1.3746628 ,
        3.06286915, 0.03730861, 0.1527219
6, 0.33126736, 0.50846386,
        0.61611025, 0.84362197, 1.4963970
2, 2.84687797, 0.03760068,
        0.15005596, 0.35538546, 0.4450898
2, 0.57916752, 0.86683154,
        1.40526676, 2.64213562]),
 'mean_test_score': array([0.58882079, 0.
65866565, 0.66984857, 0.67835435, 0.68249
781,
        0.68993669, 0.69713029, 0.6919977
5, 0.60335395, 0.67211655,
        0.68337308, 0.69318119, 0.6909387
6, 0.69511778, 0.69671805,
        0.70009407, 0.60788035, 0.6667232
7, 0.69408824, 0.69224479,
        0.70255156, 0.70246372, 0.6997935
```

```
        0.70255156, 0.70240372, 0.6997955
5, 0.70270253, 0.61404366,
        0.68095333, 0.68531186, 0.7042513
, 0.69629887, 0.70227142,
        0.70532589, 0.70322977, 0.6185954
2, 0.6886941 , 0.6953417 ,
        0.70145882, 0.69692243, 0.7045254
3, 0.70698895, 0.70630441,
        0.62716049, 0.67795005, 0.6995404
2, 0.69561302, 0.70279206,
        0.70596657, 0.70725859, 0.7119464
7, 0.64340328, 0.68395127,
        0.69667027, 0.70187992, 0.7031952
, 0.70622006, 0.70844573,
        0.71048152, 0.63717661, 0.6848458
2, 0.69623294, 0.70668566,
        0.70647437, 0.70808881, 0.7083415
1, 0.71319733, 0.6454999 ,
        0.68756015, 0.69631357, 0.7048873
3, 0.70769435, 0.70747275,
        0.7094051 , 0.71183441]),
 'mean_train_score': array([0.6197024 ,
0.71118673, 0.7180743 , 0.74052739, 0.741
93424,
        0.74644417, 0.75729199, 0.7525417
8, 0.65266872, 0.74678729,
        0.7608732 , 0.76886081, 0.7640118
1, 0.77629182, 0.77686949,
        0.77760727, 0.67172212, 0.7531281
9, 0.79098932, 0.785831  ,
        0.80120947, 0.79947703, 0.7975017
7, 0.80057116, 0.69482393,
        0.78824295, 0.80217959, 0.8139085
1, 0.80854355, 0.81404899,
        0.82188174, 0.82089658, 0.7147374
1, 0.81284206, 0.82049571,
        0.83458714, 0.8277752 , 0.8414876
```

```
1, 0.84572109, 0.84458971,
        0.73844204, 0.82489382, 0.8423218
2, 0.84886012, 0.856437  ,
        0.86276143, 0.86533837, 0.8700817
4, 0.76672316, 0.85363002,
        0.86614668, 0.87934331, 0.8765977
6, 0.88136262, 0.88435987,
        0.89007441, 0.77578522, 0.8752025
8, 0.89096766, 0.89811094,
        0.90041549, 0.900678  , 0.9024593
3, 0.90573581, 0.8019737 ,
        0.88724952, 0.90878597, 0.9132388
7, 0.91583302, 0.9210313 ,
        0.92249553, 0.92294099]),
 'param_max_depth': masked_array(data=[2,
2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3,
3, 4, 4,
                    4, 4, 4, 4, 4, 4, 5,
5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6,
                    6, 6, 6, 6, 7, 7, 7,
7, 7, 7, 7, 7, 8, 8, 8, 8, 8, 8,
                    8, 8, 9, 9, 9, 9, 9,
9, 9, 9, 10, 10, 10, 10, 10, 10,
                    10, 10],
              mask=[False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
```

```
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_n_estimators': masked_array(data=
[10, 50, 100, 150, 200, 300, 500, 1000, 1
0, 50, 100,
                    150, 200, 300, 500, 1
000, 10, 50, 100, 150, 200, 300,
                    500, 1000, 10, 50, 10
0, 150, 200, 300, 500, 1000, 10,
                    50, 100, 150, 200, 30
0, 500, 1000, 10, 50, 100, 150,
                    200, 300, 500, 1000,
10, 50, 100, 150, 200, 300, 500,
                    1000, 10, 50, 100, 15
0, 200, 300, 500, 1000, 10, 50,
                    100, 150, 200, 300, 5
00, 1000],
              mask=[False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False
```

```
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False,
                    False, False, False,
False, False, False, False, False],
        fill_value='?',
             dtype=object),
 'params': [{'max_depth': 2, 'n_estimator
s': 10},
  {'max_depth': 2, 'n_estimators': 50},
  {'max_depth': 2, 'n_estimators': 100},
  {'max_depth': 2, 'n_estimators': 150},
  {'max_depth': 2, 'n_estimators': 200},
  {'max_depth': 2, 'n_estimators': 300},
  {'max_depth': 2, 'n_estimators': 500},
  {'max_depth': 2, 'n_estimators': 1000},
  {'max_depth': 3, 'n_estimators': 10},
  {'max_depth': 3, 'n_estimators': 50},
  {'max_depth': 3, 'n_estimators': 100},
  {'max_depth': 3, 'n_estimators': 150},
  {'max_depth': 3, 'n_estimators': 200},
  {'max_depth': 3, 'n_estimators': 300},
  {'max_depth': 3, 'n_estimators': 500},
  {'max_depth': 3, 'n_estimators': 1000},
  {'max_depth': 4, 'n_estimators': 10},
  {'max_depth': 4, 'n_estimators': 50},
  {'max_depth': 4, 'n_estimators': 100},
  {'max_depth': 4, 'n_estimators': 150},
  {'max_depth': 4, 'n_estimators': 200},
  {'max_depth': 4, 'n_estimators': 300},
  {'max_depth': 4, 'n_estimators': 500},
  {'max_depth': 4, 'n_estimators': 1000},
  {'max_depth': 5, 'n_estimators': 10},
  {'max_depth': 5, 'n_estimators': 50},
  {'max_depth': 5, 'n_estimators': 100},
  {'max_depth': 5, 'n_estimators': 150},
```

```
{'max_depth': 5, 'n_estimators': 150},
{'max_depth': 5, 'n_estimators': 200},
{'max_depth': 5, 'n_estimators': 300},
{'max_depth': 5, 'n_estimators': 500},
{'max_depth': 5, 'n_estimators': 1000},
{'max_depth': 6, 'n_estimators': 10},
{'max_depth': 6, 'n_estimators': 50},
{'max_depth': 6, 'n_estimators': 100},
{'max_depth': 6, 'n_estimators': 150},
{'max_depth': 6, 'n_estimators': 200},
{'max_depth': 6, 'n_estimators': 300},
{'max_depth': 6, 'n_estimators': 500},
{'max_depth': 6, 'n_estimators': 1000},
{'max_depth': 7, 'n_estimators': 10},
{'max_depth': 7, 'n_estimators': 50},
{'max_depth': 7, 'n_estimators': 100},
{'max_depth': 7, 'n_estimators': 150},
{'max_depth': 7, 'n_estimators': 200},
{'max_depth': 7, 'n_estimators': 300},
{'max_depth': 7, 'n_estimators': 500},
{'max_depth': 7, 'n_estimators': 1000},
{'max_depth': 8, 'n_estimators': 10},
{'max_depth': 8, 'n_estimators': 50},
{'max_depth': 8, 'n_estimators': 100},
{'max_depth': 8, 'n_estimators': 150},
{'max_depth': 8, 'n_estimators': 200},
{'max_depth': 8, 'n_estimators': 300},
{'max_depth': 8, 'n_estimators': 500},
{'max_depth': 8, 'n_estimators': 1000},
{'max_depth': 9, 'n_estimators': 10},
{'max_depth': 9, 'n_estimators': 50},
{'max_depth': 9, 'n_estimators': 100},
{'max_depth': 9, 'n_estimators': 150},
{'max_depth': 9, 'n_estimators': 200},
{'max_depth': 9, 'n_estimators': 300},
{'max_depth': 9, 'n_estimators': 500},
{'max_depth': 9, 'n_estimators': 1000}
```

```
  {'max_depth': 9, 'n_estimators': 1000},
  {'max_depth': 10, 'n_estimators': 10},
  {'max_depth': 10, 'n_estimators': 50},
  {'max_depth': 10, 'n_estimators': 100},
  {'max_depth': 10, 'n_estimators': 150},
  {'max_depth': 10, 'n_estimators': 200},
  {'max_depth': 10, 'n_estimators': 300},
  {'max_depth': 10, 'n_estimators': 500},
  {'max_depth': 10, 'n_estimators': 100
0}],
 'rank_test_score': array([72, 63, 61, 5
8, 56, 49, 34, 47, 71, 60, 55, 45, 48, 4
3, 36, 31, 70,
        62, 44, 46, 26, 27, 32, 25, 69, 5
7, 52, 21, 39, 28, 18, 22, 68, 50,
        42, 30, 35, 20, 12, 15, 67, 59, 3
3, 41, 24, 17, 11,  2, 65, 54, 37,
        29, 23, 16,  6,  4, 66, 53, 40, 1
3, 14,  8,  7,  1, 64, 51, 38, 19,
         9, 10,  5,  3], dtype=int32),
 'split0_test_score': array([0.58178088,
0.65016833, 0.6691776 , 0.66912769, 0.678
89932,
        0.68030725, 0.69274553, 0.6741029
, 0.58183019, 0.65775549,
        0.67678921, 0.67432203, 0.6813034
2, 0.68705037, 0.69040159,
        0.6838306 , 0.62016076, 0.6702239
4, 0.6863711 , 0.67525222,
        0.68282434, 0.68879606, 0.6893895
3, 0.68843438, 0.61893512,
        0.68294844, 0.67474576, 0.6910933
4, 0.67768043, 0.68809201,
        0.69469599, 0.68994265, 0.6253183
5, 0.6746409 , 0.69216626,
        0.69972795, 0.67797853, 0.6890714
3, 0.69562413, 0.69651775,
```

```
3, 0.69562415, 0.69631775,
        0.64378595, 0.66769036, 0.6908477
2, 0.68629162, 0.69669534,
        0.69612068, 0.69284758, 0.6997621
4, 0.64757484, 0.69021545,
        0.68873231, 0.68416186, 0.6906166
2, 0.69532945, 0.70065831,
        0.70034432, 0.64887116, 0.6748300
3, 0.68825285, 0.69412441,
        0.6996606 , 0.70102615, 0.6976590
4, 0.70069592, 0.63764664,
        0.67697176, 0.69034194, 0.6903626
2, 0.69830805, 0.69899621,
        0.70126733, 0.70129998]),
 'split0_train_score': array([0.6123364 ,
0.70290098, 0.72478206, 0.75042222, 0.740
99342,
        0.74009789, 0.75857913, 0.7431670
3, 0.64463306, 0.738996  ,
        0.75057329, 0.75772568, 0.7691861
5, 0.77116349, 0.77327303,
        0.76615348, 0.67504785, 0.7669092
9, 0.77802802, 0.77516945,
        0.78194814, 0.79119589, 0.7915234
8, 0.79118456, 0.69556333,
        0.79684493, 0.79560856, 0.8036437
6, 0.80094507, 0.80546069,
        0.81580531, 0.81255202, 0.7158348
3, 0.80304199, 0.83008409,
        0.83392897, 0.80660042, 0.8265325
5, 0.83686425, 0.83424612,
        0.74744629, 0.81659274, 0.8369315
1, 0.83574774, 0.85222932,
        0.85722663, 0.84902276, 0.8599194
6, 0.76615716, 0.84811984,
        0.85196369, 0.86644366, 0.8679419
4, 0.8793403 , 0.87850177,
```

```
4, 0.8795405 , 0.87830177,
        0.8804516 , 0.77466428, 0.8686379
1, 0.88626033, 0.88948968,
        0.89180481, 0.89645136, 0.8946524
7, 0.89546154, 0.80319052,
        0.87499002, 0.90270343, 0.9074247
6, 0.90649041, 0.91250478,
        0.9180079 , 0.91479656]),
 'split1_test_score': array([0.58476981,
0.66494554, 0.66607062, 0.67304931, 0.679
13569,
        0.68659187, 0.69132212, 0.6928745
9, 0.62419821, 0.68199522,
        0.68983649, 0.68846267, 0.6854563
6, 0.68963801, 0.69229983,
        0.69880531, 0.59945052, 0.6627084
7, 0.69247224, 0.69200185,
        0.70401133, 0.7012189 , 0.6912626
2, 0.6988248 , 0.61043964,
        0.67955864, 0.68650118, 0.7032305
7, 0.70162202, 0.70003741,
        0.70816186, 0.70343161, 0.6238935
6, 0.68862115, 0.68975511,
        0.6881864 , 0.69167292, 0.7022916
6, 0.70600608, 0.70144422,
        0.6335545 , 0.674939  , 0.6958617
5, 0.69492814, 0.70244826,
        0.70030564, 0.70369609, 0.7065714
4, 0.6515607 , 0.66909539,
        0.69730686, 0.70522514, 0.7040356
1, 0.706222  , 0.70650049,
        0.7066017 , 0.61834929, 0.6769027
2, 0.68758428, 0.70146149,
        0.69758056, 0.70236962, 0.7063367
1, 0.71053989, 0.64574025,
        0.68694362, 0.68632825, 0.7002348
6, 0.7015191 , 0.70340665,
```

```
6, 0.7013191 , 0.70340003,
        0.70517522, 0.70912521]),
 'split1_train_score': array([0.61789047,
0.72256046, 0.7169386 , 0.73535572, 0.753
54605,
        0.75502012, 0.75710771, 0.7591162
, 0.67196055, 0.75555216,
        0.77839286, 0.76696519, 0.7536799
8, 0.77977097, 0.77945562,
        0.78494329, 0.66793378, 0.7356167
3, 0.79761492, 0.79399648,
        0.80662616, 0.80745972, 0.8010307
8, 0.80662804, 0.70238209,
        0.79495183, 0.80997244, 0.8178040
3, 0.81757034, 0.81702139,
        0.8322582 , 0.82939856, 0.7268324
6, 0.82678055, 0.82015867,
        0.82660369, 0.83302363, 0.8463484
9, 0.85388643, 0.85017131,
        0.74663841, 0.83350981, 0.8453138
2, 0.84776805, 0.86860999,
        0.86334315, 0.86905341, 0.8737448
8, 0.76870145, 0.85870056,
        0.88458586, 0.8895149 , 0.8829920
5, 0.88228126, 0.89107188,
        0.89674016, 0.781043  , 0.8819403
4, 0.89312308, 0.89951844,
        0.90546087, 0.9040045 , 0.9079782
5, 0.90889627, 0.80606137,
        0.88765803, 0.90935931, 0.9180816
6, 0.92029658, 0.927051  ,
        0.92392477, 0.92982411]),
 'split2_test_score': array([0.59991439,
0.66088466, 0.67429824, 0.6928896 , 0.689
46001,
        0.70291431, 0.70732538, 0.7090209
7, 0.60403677, 0.67660174,
```

```
7, 0.66465677, 0.67066174,
        0.68349454, 0.71676519, 0.7060601
8, 0.70866819, 0.70745527,
        0.71765136, 0.60402736, 0.6672369
7, 0.70342392, 0.7094854 ,
        0.72082469, 0.71738047, 0.7187328
7, 0.72085327, 0.61275529,
        0.68035253, 0.69469162, 0.7184340
8, 0.70959894, 0.7186894 ,
        0.71312256, 0.71631899, 0.6065715
5, 0.70282447, 0.70410551,
        0.71646461, 0.72112227, 0.7222181
5, 0.71934019, 0.72095491,
        0.60413508, 0.6912243 , 0.7119149
3, 0.70562218, 0.70923444,
        0.72147719, 0.72523696, 0.7295102
7, 0.63107183, 0.69254333,
        0.70397392, 0.71625755, 0.7149369
9, 0.71711198, 0.71818099,
        0.72450215, 0.64430872, 0.7028089
, 0.71286536, 0.72447562,
        0.7221853 , 0.72087363, 0.7210322
6, 0.72836032, 0.65311511,
        0.69876832, 0.71227378, 0.7240695
5, 0.72325962, 0.72001852,
        0.7217758 , 0.72508157]),
 'split2_train_score': array([0.62888033,
0.70809876, 0.71250224, 0.73580423, 0.731
26327,
        0.74421451, 0.75618913, 0.7553421
1, 0.64141257, 0.74581373,
        0.75365344, 0.78189157, 0.7691692
9, 0.77794099, 0.77787983,
        0.78172503, 0.67218472, 0.7568585
5, 0.79732502, 0.78832708,
        0.81505411, 0.79977549, 0.7999510
5, 0.80390087, 0.68652637,
```

```
        0.77293209, 0.80095778, 0.8202777
5, 0.80711524, 0.81966488,
        0.81758172, 0.82073917, 0.7015449
4, 0.80870363, 0.81124438,
        0.84322877, 0.84370156, 0.8515817
9, 0.84641259, 0.84935171,
        0.72124143, 0.8245789 , 0.8447201
2, 0.86306457, 0.84847169,
        0.86771452, 0.87793894, 0.8765808
8, 0.76531088, 0.85406966,
        0.86189048, 0.88207137, 0.8788593
, 0.88246629, 0.88350597,
        0.89303149, 0.77164839, 0.8750294
9, 0.89351957, 0.90532471,
        0.90398079, 0.90157814, 0.9047472
6, 0.91284963, 0.79666919,
        0.89910053, 0.91429517, 0.9142101
7, 0.92071208, 0.92353812,
        0.92555392, 0.92420232]),
 'std_fit_time': array([2.64715303e-03,
9.07475431e-03, 7.18870753e-03, 1.2655948
6e-02,
        2.10100628e-02, 1.14673339e-02,
2.51596521e-02, 3.96998066e-02,
        3.26193969e-04, 7.59155115e-03,
1.12788730e-02, 2.39795620e-02,
        8.00957057e-03, 3.99864313e-03,
4.01201582e-02, 1.61299195e-02,
        3.83993115e-03, 9.87455960e-03,
1.64216850e-02, 2.58659507e-03,
        9.06536296e-03, 1.17790451e-02,
7.33845391e-03, 6.79358374e-02,
        4.03671109e-03, 1.77380721e-02,
6.25647003e-03, 2.36600040e-02,
        3.90927926e-02, 3.33988776e-02,
6.97142306e-02, 5.23595545e-02,
```

6.97142306e-02, 3.25595545e-02,
        2.95221995e-03, 4.10157542e-03,
1.50602167e-02, 1.59042808e-02,
        3.25811906e-02, 1.66460728e-02,
4.65512352e-02, 1.23844968e-01,
        9.18078490e-04, 1.51946516e-02,
2.49366239e-02, 3.13502139e-02,
        7.41337810e-02, 5.78462956e-02,
4.73494123e-02, 6.89234036e-02,
        4.47856843e-03, 2.69844177e-02,
1.34704536e-02, 1.87724383e-02,
        4.08328512e-02, 4.68954065e-02,
4.73528658e-02, 1.26548000e-01,
        4.92569853e-03, 2.53268884e-02,
9.67957401e-03, 1.59737380e-02,
        3.34046039e-02, 4.40035813e-02,
8.76916958e-02, 1.27832859e-01,
        4.84454748e-03, 1.90775956e-02,
1.35033527e-02, 3.09696356e-02,
        6.78122271e-02, 6.09879690e-02,
1.02082379e-01, 1.39141722e+00]),
 'std_score_time': array([2.31533954e-03,
1.87778196e-02, 2.23493414e-02, 3.7535201
6e-02,
        2.34093295e-02, 1.09606769e-01,
1.32221719e-01, 3.11397212e-01,
        3.28357689e-03, 3.18398165e-03,
1.05771791e-02, 5.49829019e-02,
        6.34085045e-02, 4.83634211e-03,
1.64257704e-01, 2.80936831e-01,
        1.47878257e-04, 1.59010467e-03,
7.37538218e-03, 2.45674873e-03,
        6.09437724e-02, 7.77065815e-02,
1.75146694e-01, 2.74853691e-01,
        9.34147265e-05, 2.02988426e-02,
1.31577919e-03, 4.80128232e-02,
        6.03478881e-02, 3.37757884e-02

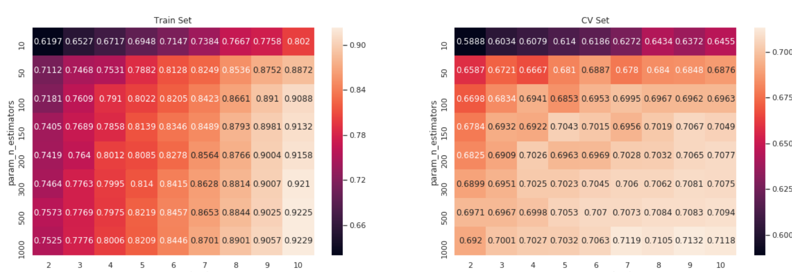
```
1.50674594e-01, 2.54117450e-01,
        6.99467515e-04, 2.04906093e-02,
1.09525340e-03, 3.45255127e-02,
        6.66886140e-03, 8.35053411e-02,
8.13368663e-02, 2.76604121e-01,
        2.33733423e-04, 6.62944312e-03,
2.05250300e-03, 4.47530653e-02,
        6.06200888e-02, 6.10288656e-02,
1.71092138e-01, 1.40942530e-01,
        2.49213000e-05, 1.27235954e-02,
1.52114375e-02, 4.67758759e-02,
        3.80117529e-02, 9.56827596e-02,
1.40946506e-02, 2.79700178e-01,
        1.72070288e-03, 6.90701124e-03,
3.20725822e-02, 3.70980860e-02,
        5.57961148e-02, 1.45023201e-02,
1.57268827e-01, 8.91540278e-02,
        1.77502014e-03, 4.42153543e-03,
3.88392740e-03, 1.56442295e-02,
        1.37234879e-02, 2.89111286e-02,
6.93163493e-03, 4.49343138e-01]),
 'std_test_score': array([0.00793784, 0.0
0623366, 0.00339214, 0.01040079, 0.004923
41,
        0.00952753, 0.0072316 , 0.0142687
2, 0.01730405, 0.01039187,
        0.00532742, 0.01764566, 0.0108248
7, 0.00963859, 0.00763097,
        0.01383731, 0.00888325, 0.0030897
1, 0.00705494, 0.01397669,
        0.01554788, 0.01170269, 0.0134124
4, 0.01351601, 0.00358602,
        0.00144766, 0.00818618, 0.0111851
2, 0.01356342, 0.01259082,
        0.00778531, 0.01076904, 0.0085210
9, 0.01150601, 0.00627396,
```

```
9, 0.01130001, 0.00027390,
        0.01160882, 0.01800025, 0.0136239
6, 0.00970695, 0.01055177,
        0.01680695, 0.00984078, 0.0089854
1, 0.00790651, 0.00512484,
        0.01109873, 0.01346072, 0.0127254
6, 0.00886923, 0.0105476 ,
        0.00623862, 0.01331481, 0.0099465
2, 0.00889268, 0.00728464,
        0.01023686, 0.0134426 , 0.0127285
5, 0.01176275, 0.01292976,
        0.01114047, 0.00905584, 0.0096468
1, 0.0114492 , 0.00631726,
        0.00890908, 0.01140267, 0.0141485
8, 0.01108288, 0.00905112,
        0.00889075, 0.00989599]),
 'std_train_score': array([0.00687448, 0.
00831767, 0.00507713, 0.0069991 , 0.00912
12 ,
        0.00629268, 0.00098438, 0.0068056
5, 0.01370459, 0.00679399,
        0.01245193, 0.00995633, 0.0073057
1, 0.00370243, 0.00262319,
        0.00820493, 0.00292267, 0.0130446
1, 0.00916579, 0.00788615,
        0.0140477 , 0.00664303, 0.0042502
1, 0.00673006, 0.00649415,
        0.01085396, 0.00592733, 0.0073282
, 0.00686197, 0.00616799,
        0.00737302, 0.00687847, 0.0103527
1, 0.0101234 , 0.00769497,
        0.0068031 , 0.01559451, 0.0107884
9, 0.00696645, 0.00732167,
        0.01216714, 0.00690995, 0.0038192
2, 0.01117875, 0.00874323,
        0.00430138, 0.01209373, 0.0072784
9, 0.00144089, 0.00433073
```

```
         0.01365376, 0.0096143 , 0.0063488
8, 0.00143199, 0.00516713,
         0.00697077, 0.00391638, 0.0054320
8, 0.00333252, 0.00654079,
         0.00611858, 0.00314856, 0.0056756
8, 0.00744212, 0.00392969,
         0.00984731, 0.00474964, 0.0044045
4, 0.0066084 , 0.00619738,
         0.00324219, 0.00619946])}
```

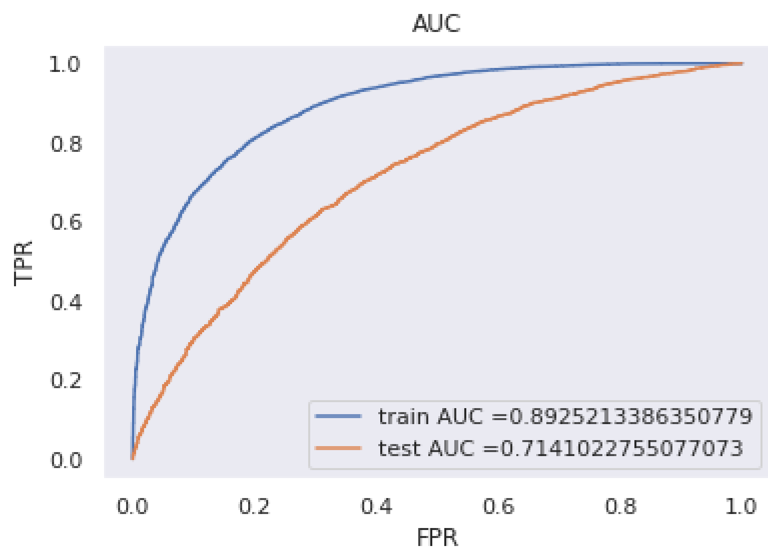
In [137]:

```
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(gs.cv_results_).grou
pby(['param_n_estimators', 'param_max_depth']).
max().unstack()[['mean_test_score', 'mean_train
_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot
= True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot
= True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [138]:
```
gs_results.best_params_
```
Out[138]:
```
{'max_depth': 9, 'n_estimators': 1000}
```
In [0]:
```
max_d = gs_results.best_params_['max_depth']
n_est = gs_results.best_params_['n_estimators']
```
In [0]:
```
def pred_prob(clf, data):
    y_pred = []
    y_pred = clf.predict_proba(data)[:,1]
    return y_pred
```
In [141]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
model = RandomForestClassifier(max_depth = max_d, n_estimators = n_est)

model.fit(X_train,y_train)

y_train_pred = pred_prob(model,X_train)
y_test_pred = pred_prob(model,X_test)

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.close
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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [0]:

```
# we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshold, fpr, tpr):
    t = threshold[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))
    return t

def predict_with_best_t(proba, threshold):
    predictions = []
```

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

In [143]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give
 the least fpr
best_t = find_best_threshold(tr_thresholds, tra
in_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_be
st_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.648988
8795417483 for threshold 0.84
Train confusion matrix
[[ 2493   602]
 [ 3304 13701]]
```

In [144]:

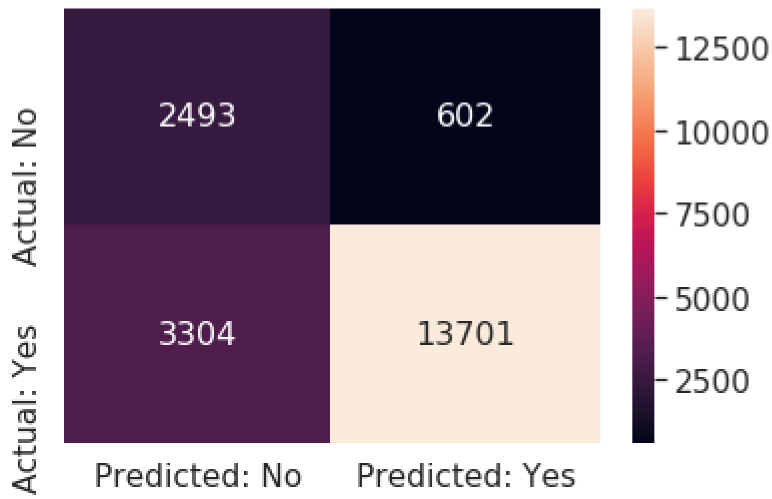
```
#plotting confusion matrix using seaborn's heat
map
# https://stackoverflow.com/questions/35572000/
how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confus
ion_matrix(y_train, predict_with_best_t(y_train
_pred, best_t)), ['Actual: No','Actual: Yes'],[
'Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
```

```
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[144]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d68e20908>
```



In [145]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 969  556]
 [2668 5707]]
```

In [146]:

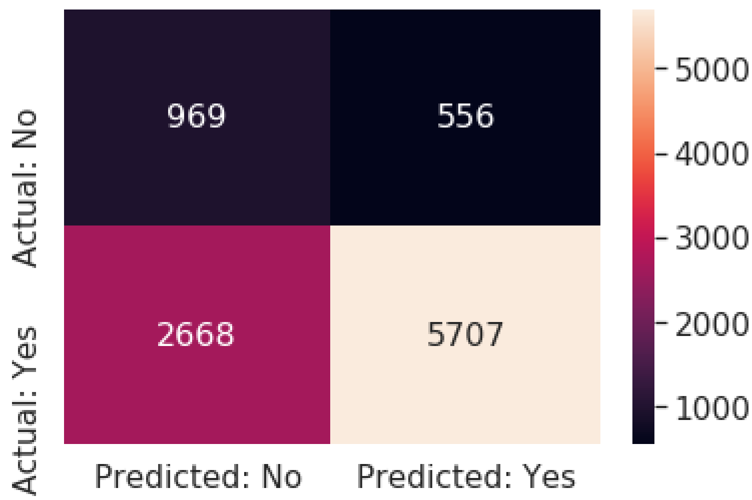
```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
```

```
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[146]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d68c27400>
```



SET 2 categorical (with response coding), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

In [0]:

```
# Please write all the code with proper documentation
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_train = hstack((cat_0_train_normalized, cat_1_train_normalized, subcat_0_train_normalized, subcat_1_train_normalized, state_0_train_normalized, state_1_train_normalized, grade_0_train_normalized, grade_1_train_normalized, prefix_0_tr
```

```
ain_normalized, prefix_1_train_normalized, pric
e_normalized_train, quantity_normalized_train,
previously_posted_projects_normalized_train, ti
tle_word_count_normalized_train, essay_word_cou
nt_normalized_train, sent_pos_train, sent_neg_t
rain, sent_neu_train, sent_compound_train, trai
n_title_tfidf, train_essay_tfidf)).tocsr()
X_test =  hstack((cat_0_test_normalized, cat_1_
test_normalized, subcat_0_test_normalized, subc
at_1_test_normalized, state_0_test_normalized,
state_1_test_normalized, grade_0_test_normalize
d, grade_1_test_normalized, prefix_0_test_norma
lized, prefix_1_test_normalized, price_normaliz
ed_test, quantity_normalized_test, previously_p
osted_projects_normalized_test, title_word_coun
t_normalized_test, essay_word_count_normalized_
test, sent_pos_test, sent_neg_test, sent_neu_te
st, sent_compound_test, test_title_tfidf, test_
essay_tfidf)).tocsr()
```

In [148]:

```
print(X_train.shape)
print(X_test.shape)
```

```
(20100, 9516)
(9900, 9516)
```

In [152]:

```
# https://medium.com/@erikgreenj/k-neighbors-cl
assifier-with-gridsearchcv-basics-3c445ddeb657

from sklearn.model_selection import GridSearchC
V
from sklearn.ensemble import RandomForestClassi
fier

rf = RandomForestClassifier()
```

```
grid_params = {'n_estimators': [10, 50, 100, 150, 200, 300, 500, 1000], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 10]}

gs = GridSearchCV(rf, grid_params, cv=3, scoring='roc_auc',n_jobs=-1,return_train_score=True)
gs_results = gs.fit(X_train, y_train)
print(gs_results.best_score_)
print(gs_results.best_estimator_)
print(gs_results.best_params_)
```

```
0.6963252641049676
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=9, max_features='auto', max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=1000,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
{'max_depth': 9, 'n_estimators': 1000}
```

In [153]:

```
#Output of GridSearchCV
print('Best score: ',gs_results.best_score_)
print('k value with best score: ',gs_results.best_params_)
print('='*75)
print('Train AUC scores')
```

```python
print(gs.cv_results_['mean_train_score'])
print('CV AUC scores')
print(gs.cv_results_['mean_test_score'])
```

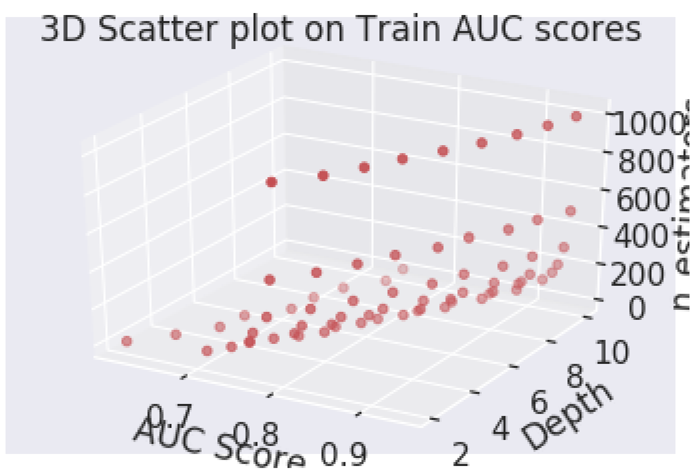
```
Best score:  0.6963252641049676
k value with best score:  {'max_depth':
9, 'n_estimators': 1000}
=========================================
==================================
Train AUC scores
[0.61677867 0.7122381  0.74094067 0.76218
489 0.76608163 0.78234209
 0.78651346 0.79076986 0.6490126  0.73635
837 0.76483742 0.78906211
 0.79742746 0.80761064 0.81476414 0.82299
959 0.67594832 0.76932763
 0.79916522 0.8076891  0.81834405 0.83221
149 0.83689881 0.84419568
 0.68031669 0.787986   0.81786065 0.82692
914 0.84287273 0.85381373
 0.85610457 0.86280304 0.71173569 0.81365
435 0.84330106 0.86284464
 0.8660914  0.87555254 0.88124344 0.88456
929 0.71405107 0.8372557
 0.86880691 0.87512642 0.88881704 0.88924
86  0.89382451 0.90482973
 0.72588455 0.8544873  0.88902836 0.90169
671 0.90239734 0.91186362
 0.91648398 0.9219205  0.75350439 0.87642
947 0.90752149 0.91260333
 0.92189626 0.92381083 0.92786558 0.93439
443 0.75126437 0.8881536
 0.91607283 0.92671765 0.93277017 0.93872

555 0.94426418 0.94494713]
CV AUC scores
[0.57935004 0.64584145 0.65605736 0.66736
```

```
[0.57955004 0.64504145 0.65605756 0.66756
477 0.66345019 0.68310108
 0.68011472 0.68223866 0.60523083 0.63956
02  0.65575191 0.6702175
 0.67765116 0.6811316  0.68210226 0.68791
182 0.62353178 0.65542802
 0.66967051 0.67301814 0.67495544 0.68418
083 0.6884634  0.6869463
 0.62103748 0.65632425 0.65783937 0.67833
257 0.68662551 0.68488573
 0.6898536  0.69384059 0.62749059 0.66009
009 0.66048905 0.68097767
 0.68239246 0.68594469 0.69022666 0.69299
446 0.61142743 0.66851549
 0.67617161 0.67964094 0.68650298 0.68743
915 0.69051431 0.69484284
 0.62559273 0.66432667 0.67792814 0.68589
832 0.68445587 0.69071491
 0.69215575 0.69349458 0.63378988 0.66078
232 0.68008104 0.68416389
 0.6883958  0.6939581  0.69185484 0.69632
526 0.62194069 0.66629405
 0.67842044 0.68595033 0.68280975 0.68786
051 0.69052289 0.69598699]
```

In [154]:

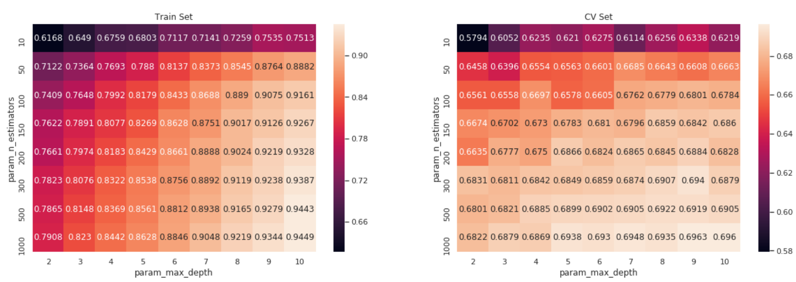
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

import numpy as np

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

g1 = list(gs.cv_results_['mean_train_score'])
#Train AUC Score
g2 = [2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,4,4,4,4,
```

```
4,4,4,4,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,7,7,7,
7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,10,1
0,10,10,10,10,10,10] #Depth
g3 = [10, 50, 100, 150, 200, 300, 500, 1000,10,
50, 100, 150, 200, 300, 500, 1000,10, 50, 100,
150, 200, 300, 500, 1000,10, 50, 100, 150, 200,
300, 500, 1000,10, 50, 100, 150, 200, 300, 500,
1000,10, 50, 100, 150, 200, 300, 500, 1000,10,
50, 100, 150, 200, 300, 500, 1000,10, 50, 100,
150, 200, 300, 500, 1000,10, 50, 100, 150, 200,
300, 500, 1000] #n_estimators




ax.scatter(g1, g2, g3, c='r', marker='o')

ax.set_xlabel('AUC Score')
ax.set_ylabel('Depth')
ax.set_zlabel('n_estimators')

plt.title('3D Scatter plot on Train AUC scores'
)
plt.show()
```



In [155]:

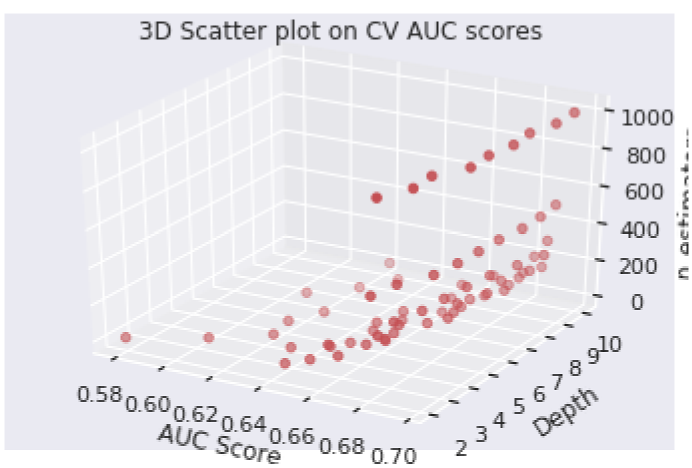
```python
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(gs.cv_results_).grou
pby(['param_n_estimators', 'param_max_depth']).
max().unstack()[['mean_test_score', 'mean_train
_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot
= True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot
= True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [156]:

```python
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt




fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

g1 = list(gs.cv_results_['mean_test_score'])
#Train AUC Score
g2 = [2,2,2,2,2,2,2,2,3,3,3,3,3,3,3,3,4,4,4,4,
4,4,4,4,5,5,5,5,5,5,5,5,6,6,6,6,6,6,6,6,7,7,7,
7,7,7,7,7,8,8,8,8,8,8,8,8,9,9,9,9,9,9,9,9,10,1
0,10,10,10,10,10,10] #Depth
```

```
g3 = [10, 50, 100, 150, 200, 300, 500, 1000,10,
50, 100, 150, 200, 300, 500, 1000,10, 50, 100,
150, 200, 300, 500, 1000,10, 50, 100, 150, 200,
300, 500, 1000,10, 50, 100, 150, 200, 300, 500,
1000,10, 50, 100, 150, 200, 300, 500, 1000,10,
50, 100, 150, 200, 300, 500, 1000,10, 50, 100,
150, 200, 300, 500, 1000,10, 50, 100, 150, 200,
300, 500, 1000] #n_estimators




ax.scatter(g1, g2, g3, c='r', marker='o')

ax.set_xlabel('AUC Score')
ax.set_ylabel('Depth')
ax.set_zlabel('n_estimators')

plt.title('3D Scatter plot on CV AUC scores')
plt.show()
```



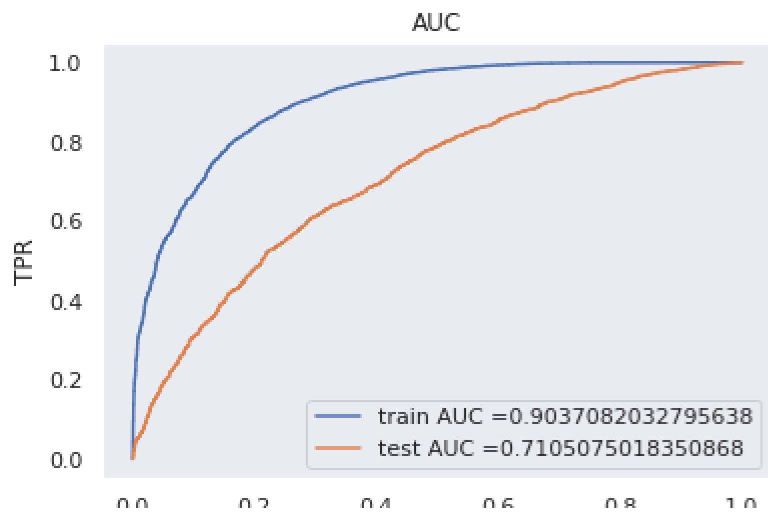
In [157]:

```
gs_results.best_params_
```

Out[157]: {'max_depth': 9, 'n_estimators': 1000}

In [0]:
```
max_d = gs_results.best_params_['max_depth']
n_est = gs_results.best_params_['n_estimators']
```

In [159]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
model = RandomForestClassifier(max_depth = max_d, n_estimators = n_est)

model.fit(X_train,y_train)

y_train_pred = pred_prob(model,X_train)
y_test_pred = pred_prob(model,X_test)

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.close
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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [160]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give
 the least fpr
best_t = find_best_threshold(tr_thresholds, tra
in_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_be
st_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.670822
3324984242 for threshold 0.84
Train confusion matrix
[[ 2454   641]
 [ 2618 14387]]
```

In [161]:

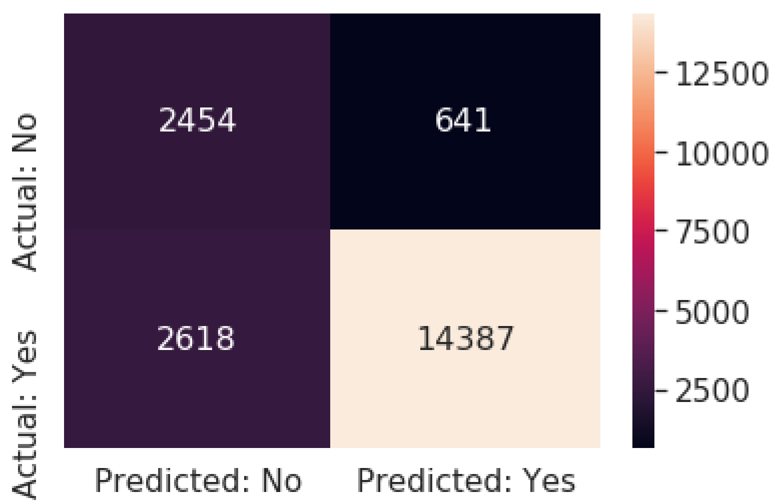
```
#plotting confusion matrix using seaborn's heat
map
# https://stackoverflow.com/questions/35572000/
how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confus
```

```
ion_matrix(y_train, predict_with_best_t(y_train
_pred, best_t)), ['Actual: No','Actual: Yes'],[
'Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=Tr
ue,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[161]:

```
<matplotlib.axes._subplots.AxesSubplot at
0x7f7d68b96198>
```



In [162]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_bes
t_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 835  690]
 [2133 6242]]
```

In [163]:

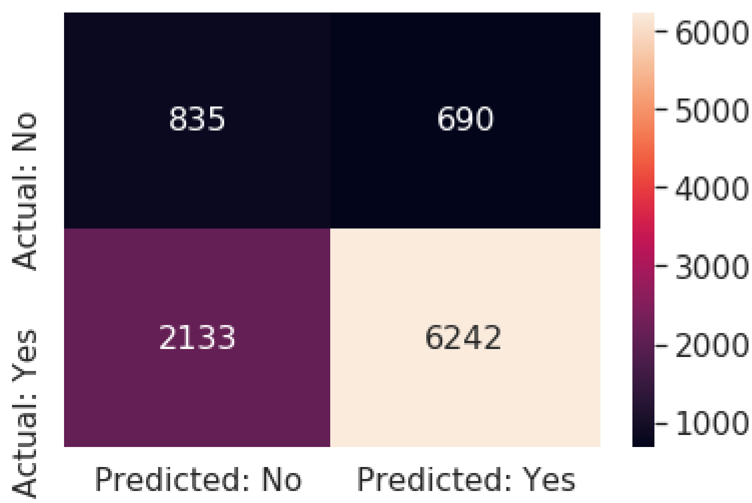
```
print("Test data confusion matrix")


confusion_matrix_df_test = pd.DataFrame(confusi
```

```
on_matrix(y_test, predict_with_best_t(y_test_pr
ed, best_t)), ['Actual: No','Actual: Yes'],['Pr
edicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=Tru
e,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[163]:

```
<matplotlib.axes._subplots.AxesSubplot at
0x7f7d68d66198>
```



2.4.3 Applying Random Forests on AVG W2V, SET 3

In [0]:

```
train_avg_w2v_essays_np = np.array(train_avg_w2
v_essays)
train_avg_w2v_titles_np = np.array(train_avg_w2
v_titles)
test_avg_w2v_essays_np  = np.array(test_avg_w2v
_essays)
test_avg_w2v_titles_np  = np.array(test_avg_w2v
_titles)
```

In [165]:

```
print(cat_0_train_normalized.shape)
print(cat_1_train_normalized.shape)
print(subcat_0_train_normalized.shape)
print(subcat_1_train_normalized.shape)
print(state_0_train_normalized.shape)
print(state_1_train_normalized.shape)
print(grade_0_train_normalized.shape)
print(grade_1_train_normalized.shape)
print(prefix_0_train_normalized.shape)
print(prefix_1_train_normalized.shape)
print(price_normalized_train.shape)
print(quantity_normalized_train.shape)
print(previously_posted_projects_normalized_tra
in.shape)
print(title_word_count_normalized_train.shape)
print(essay_word_count_normalized_train.shape)
print(sent_pos_train.shape)
print(sent_neg_train.shape)
print(sent_neu_train.shape)
print(sent_compound_train.shape)
print(train_avg_w2v_essays_np.shape)
print(train_avg_w2v_titles_np.shape)
```

```
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
```

```
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 1)
(20100, 300)
(20100, 300)
```

In [0]:

```
#https://blog.csdn.net/w55100/article/details/90369779
# if you use hstack without converting it into to a sparse matrix first,
#it shows an error: blocks must be 2-D

from scipy.sparse import coo_matrix, hstack
tr1 = coo_matrix(cat_0_train_normalized)
tr2 = coo_matrix(cat_1_train_normalized)
tr3 = coo_matrix(subcat_0_train_normalized)
tr4 = coo_matrix(subcat_1_train_normalized)
tr5 = coo_matrix(state_0_train_normalized)
tr6 = coo_matrix(state_1_train_normalized)
tr7 = coo_matrix(grade_0_train_normalized)
tr8 = coo_matrix(grade_1_train_normalized)
tr9 = coo_matrix(prefix_0_train_normalized)
tr10 = coo_matrix(prefix_1_train_normalized)
tr11 = coo_matrix(price_normalized_train)
tr12 = coo_matrix(quantity_normalized_train)
tr13 = coo_matrix(previously_posted_projects_normalized_train)
tr14 = coo_matrix(title_word_count_normalized_train)
tr15 = coo_matrix(essay_word_count_normalized_train)
```

```
tr16 = coo_matrix(sent_pos_train)
tr17 = coo_matrix(sent_neg_train)
tr18 = coo_matrix(sent_neu_train)
tr19 = coo_matrix(sent_compound_train)
tr20 = coo_matrix(train_avg_w2v_essays_np)
tr21 = coo_matrix(train_avg_w2v_titles_np)
```

In [0]:

```
X_train = hstack([tr1,tr2,tr3,tr4,tr5,tr6,tr7,tr8,tr9,tr10,tr11,tr12,tr13,tr14,tr15,tr16,tr17,tr18,tr19,tr20,tr21]).tocsr()
```

In [0]:

```
te1 = coo_matrix(cat_0_test_normalized)
te2 = coo_matrix(cat_1_test_normalized)
te3 = coo_matrix(subcat_0_test_normalized)
te4 = coo_matrix(subcat_1_test_normalized)
te5 = coo_matrix(state_0_test_normalized)
te6 = coo_matrix(state_1_test_normalized)
te7 = coo_matrix(grade_0_test_normalized)
te8 = coo_matrix(grade_1_test_normalized)
te9 = coo_matrix(prefix_0_test_normalized)
te10 = coo_matrix(prefix_1_test_normalized)
te11 = coo_matrix(price_normalized_test)
te12 = coo_matrix(quantity_normalized_test)
te13 = coo_matrix(previously_posted_projects_normalized_test)
te14 = coo_matrix(title_word_count_normalized_test)
te15 = coo_matrix(essay_word_count_normalized_test)
te16 = coo_matrix(sent_pos_test)
te17 = coo_matrix(sent_neg_test)
te18 = coo_matrix(sent_neu_test)
te19 = coo_matrix(sent_compound_test)
te20 = coo_matrix(test_avg_w2v_essays_np)
te21 = coo_matrix(test_avg_w2v_titles_np)
```

In [0]:

```
X_test = hstack([te1,te2,te3,te4,te5,te6,te7,te8,te9,te10,te11,te12,te13,te14,te15,te16,te17,te18,te19,te20,te21]).tocsr()
```

In [170]:

```
print(X_train.shape)
print(X_test.shape)
```

```
(20100, 619)
(9900, 619)
```

In [171]:

```
from sklearn.model_selection import GridSearchCV
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV

rf = RandomForestClassifier()

grid_params = {'n_estimators': [10, 50, 100, 150, 200, 300, 500, 1000], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 10]}

rs = RandomizedSearchCV(rf,grid_params ,cv=3, scoring='roc_auc',n_jobs=-1,return_train_score=True)
rs.fit(X_train, y_train)
```

Out[171]:

```
RandomizedSearchCV(cv=3, error_score='raise-deprecating',
                   estimator=RandomForestClassifier(bootstrap=True,
                                                    class_weight=None,
                                                    criterion='gini',
```

```
max_depth=None,

max_features='auto',

max_leaf_nodes=None,

min_impurity_decrease=0.0,

min_impurity_split=None,

min_samples_leaf=1,

min_samples_split=2,

min_weight_fraction_leaf=0.0,

n_estimators='warn',

n_jobs=None,

oob_score=False,

random_state=None,

verbose=0,

warm_start=False),
                   iid='warn', n_iter=10,
n_jobs=-1,
                   param_distributions=
{'max_depth': [2, 3, 4, 5, 6, 7, 8, 9,

10],
```

```
'n_estimators': [10, 50, 100, 150, 200,

300, 500, 1000]},
                    pre_dispatch='2*n_job
s', random_state=None, refit=True,
                    return_train_score=Tru
e, scoring='roc_auc', verbose=0)
```

In [172]:

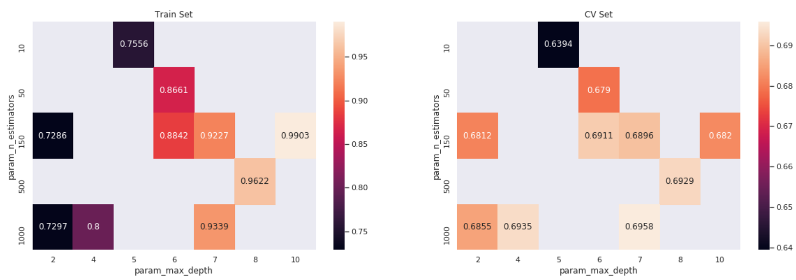
```
print('Best score: ',rs.best_score_)
print('k value with best score: ',rs.best_param
s_)
print('='*75)
print('Train AUC scores')
print(rs.cv_results_['mean_train_score'])
print('CV AUC scores')
print(rs.cv_results_['mean_test_score'])
```

```
Best score:  0.6958344814738235
k value with best score:  {'n_estimator
s': 1000, 'max_depth': 7}
=========================================
==================================
Train AUC scores
[0.72973238 0.75558022 0.9338576  0.96223
154 0.88420485 0.72862269
 0.86607493 0.99031803 0.92274474 0.79995
046]
CV AUC scores
[0.68545908 0.63935916 0.69583448 0.69286
449 0.69113485 0.68117059
 0.67901794 0.68199784 0.68957848 0.69352
643]
```

In [173]:

```
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(rs.cv_results_).grou
pby(['param_n_estimators', 'param_max_depth']).
```

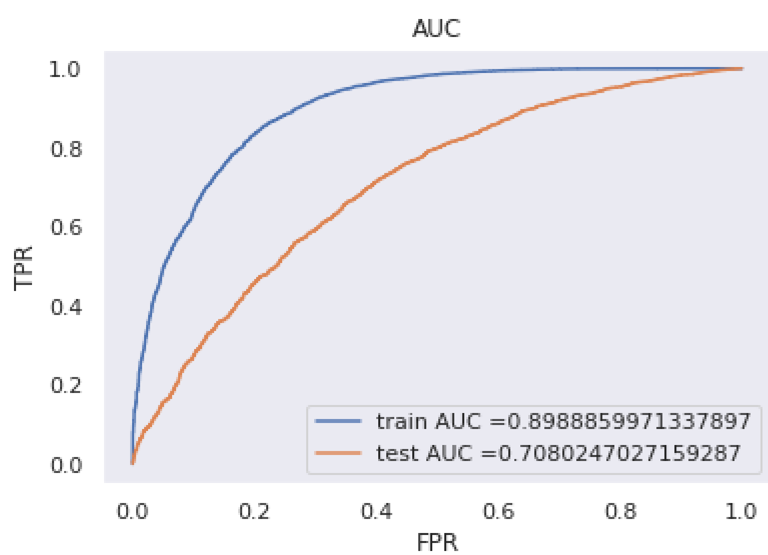
```
max().unstack()[['mean_test_score', 'mean_train_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [0]:
```
max_d = rs.best_params_['max_depth']
n_est = rs.best_params_['n_estimators']
```

In [175]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
model = RandomForestClassifier(max_depth = max_d, n_estimators = n_est)

model.fit(X_train,y_train)

y_train_pred = pred_prob(model,X_train)
y_test_pred = pred_prob(model,X_test)

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.close
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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [176]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give
 the least fpr
best_t = find_best_threshold(tr_thresholds, tra
in_fpr, train_tpr)
print("Train confusion matrix")
```

```python
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.6704904715376405 for threshold 0.832
Train confusion matrix
[[ 2433   662]
 [ 2501 14504]]
```

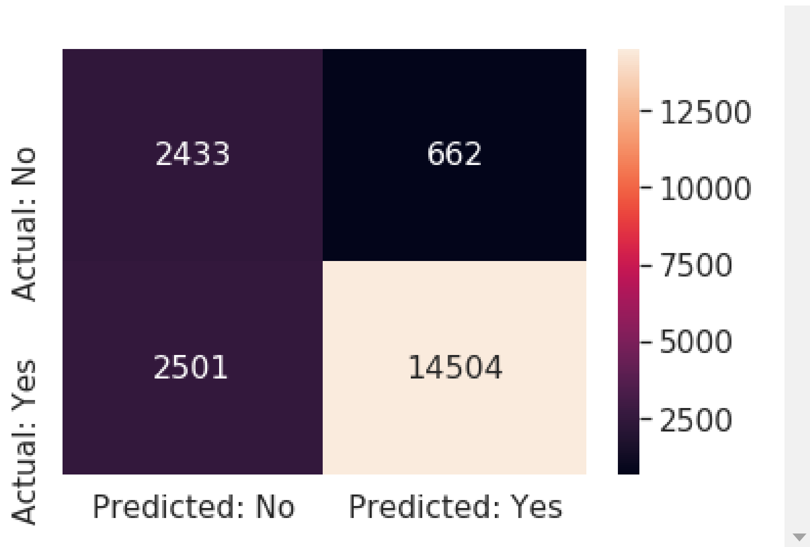
In [177]:

```python
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[177]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d68de6eb8>
```



In [178]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 910  615]
 [2352 6023]]
```

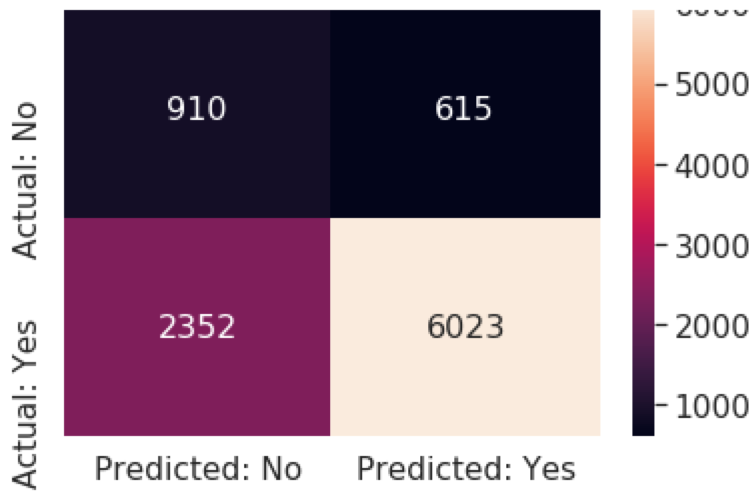
In [179]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[179]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d6884a9b0>
```



2.4.4 Applying Random Forests on TFIDF W2V, SET 4

In [0]:
```
train_tfidf_w2v_essays_np = np.array(train_tfidf_w2v_essays)
train_tfidf_w2v_titles_np = np.array(train_tfidf_w2v_titles)
test_tfidf_w2v_essays_np  = np.array(test_tfidf_w2v_essays)
test_tfidf_w2v_titles_np  = np.array(test_tfidf_w2v_titles)
```

In [0]:
```
#https://blog.csdn.net/w55100/article/details/90369779
# if you use hstack without converting it into to a sparse matrix first,
#it shows an error: blocks must be 2-D

from scipy.sparse import coo_matrix, hstack
tr1 = coo_matrix(cat_0_train_normalized)
tr2 = coo_matrix(cat_1_train_normalized)
tr3 = coo_matrix(subcat_0_train_normalized)
tr4 = coo_matrix(subcat_1_train_normalized)
```

```
tr5 = coo_matrix(state_0_train_normalized)
tr6 = coo_matrix(state_1_train_normalized)
tr7 = coo_matrix(grade_0_train_normalized)
tr8 = coo_matrix(grade_1_train_normalized)
tr9 = coo_matrix(prefix_0_train_normalized)
tr10 = coo_matrix(prefix_1_train_normalized)
tr11 = coo_matrix(price_normalized_train)
tr12 = coo_matrix(quantity_normalized_train)
tr13 = coo_matrix(previously_posted_projects_no
rmalized_train)
tr14 = coo_matrix(title_word_count_normalized_t
rain)
tr15 = coo_matrix(essay_word_count_normalized_t
rain)
tr16 = coo_matrix(sent_pos_train)
tr17 = coo_matrix(sent_neg_train)
tr18 = coo_matrix(sent_neu_train)
tr19 = coo_matrix(sent_compound_train)
tr20 = coo_matrix(train_tfidf_w2v_essays_np)
tr21 = coo_matrix(train_tfidf_w2v_titles_np)
```

In [0]:

```
X_train = hstack([tr1,tr2,tr3,tr4,tr5,tr6,tr7,t
r8,tr9,tr10,tr11,tr12,tr13,tr14,tr15,tr16,tr17,
tr18,tr19,tr20,tr21]).tocsr()
```

In [0]:

```
te1 = coo_matrix(cat_0_test_normalized)
te2 = coo_matrix(cat_1_test_normalized)
te3 = coo_matrix(subcat_0_test_normalized)
te4 = coo_matrix(subcat_1_test_normalized)
te5 = coo_matrix(state_0_test_normalized)
te6 = coo_matrix(state_1_test_normalized)
te7 = coo_matrix(grade_0_test_normalized)
te8 = coo_matrix(grade_1_test_normalized)
te9 = coo_matrix(prefix_0_test_normalized)
te10 = coo_matrix(prefix_1_test_normalized)
```

```
te11 = coo_matrix(price_normalized_test)
te12 = coo_matrix(quantity_normalized_test)
te13 = coo_matrix(previously_posted_projects_no
rmalized_test)
te14 = coo_matrix(title_word_count_normalized_t
est)
te15 = coo_matrix(essay_word_count_normalized_t
est)
te16 = coo_matrix(sent_pos_test)
te17 = coo_matrix(sent_neg_test)
te18 = coo_matrix(sent_neu_test)
te19 = coo_matrix(sent_compound_test)
te20 = coo_matrix(test_tfidf_w2v_essays_np)
te21 = coo_matrix(test_tfidf_w2v_titles_np)
```

In [0]:

```
X_test = hstack([te1,te2,te3,te4,te5,te6,te7,te
8,te9,te10,te11,te12,te13,te14,te15,te16,te17,t
e18,te19,te20,te21]).tocsr()
```

In [185]:

```
print(X_train.shape)
print(X_test.shape)
```

```
(20100, 619)
(9900, 619)
```

In [186]:

```
from sklearn.model_selection import GridSearchC
V
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedS
earchCV

rf = RandomForestClassifier()

grid_params = {'n_estimators': [10, 50, 100, 15
0, 200, 300, 500, 1000], 'max_depth':[2, 3, 4,
```

```
5, 6, 7, 8, 9, 10]}

rs = RandomizedSearchCV(rf,grid_params ,cv=3, s
coring='roc_auc',n_jobs=-1,return_train_score=T
rue)
rs.fit(X_train, y_train)
```

Out[186]:

```
RandomizedSearchCV(cv=3, error_score='rai
se-deprecating',
                   estimator=RandomForest
Classifier(bootstrap=True,

class_weight=None,

criterion='gini',

max_depth=None,

max_features='auto',

max_leaf_nodes=None,

min_impurity_decrease=0.0,

min_impurity_split=None,

min_samples_leaf=1,

min_samples_split=2,

min_weight_fraction_leaf=0.0,

n_estimators='warn',

n_jobs=None,
```

```
oob_score=False,

random_state=None,

verbose=0,

warm_start=False),
                   iid='warn', n_iter=10,
n_jobs=-1,
                   param_distributions=
{'max_depth': [2, 3, 4, 5, 6, 7, 8, 9,

10],

'n_estimators': [10, 50, 100, 150, 200,

300, 500, 1000]},
                   pre_dispatch='2*n_job
s', random_state=None, refit=True,
                   return_train_score=Tru
e, scoring='roc_auc', verbose=0)
```

In [187]:

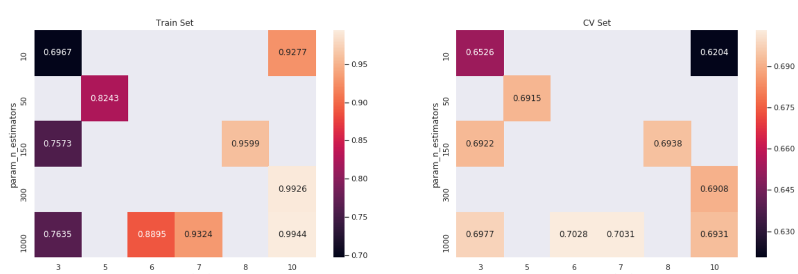
```
print('Best score: ',rs.best_score_)
print('k value with best score: ',rs.best_param
s_)
print('='*75)
print('Train AUC scores')
print(rs.cv_results_['mean_train_score'])
print('CV AUC scores')
print(rs.cv_results_['mean_test_score'])
```

```
Best score:  0.7030672667359812
k value with best score:  {'n_estimator
s': 1000, 'max_depth': 7}
```

```
========================================
==================================
Train AUC scores
[0.82433106 0.93237877 0.88948088 0.95986
049 0.99256362 0.76352589
 0.92770039 0.75733928 0.69668132 0.99439
491]
CV AUC scores
[0.69145487 0.70306727 0.70281972 0.69378
013 0.69077295 0.69772586
 0.6204351  0.69221983 0.65261872 0.69314
499]
```

In [188]:

```python
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(rs.cv_results_).grou
pby(['param_n_estimators', 'param_max_depth']).
max().unstack()[['mean_test_score', 'mean_train
_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot
= True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot
= True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```

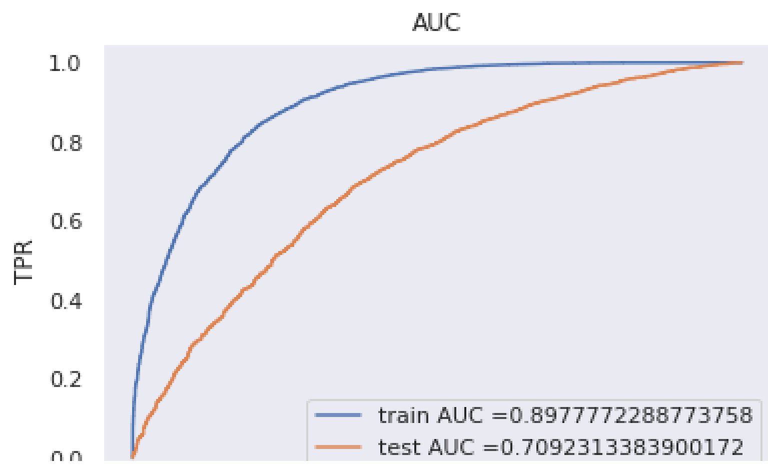


In [0]:

```
max_d = rs.best_params_['max_depth']
n_est = rs.best_params_['n_estimators']
```

In [190]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
model = RandomForestClassifier(max_depth = max_d, n_estimators = n_est)

model.fit(X_train,y_train)

y_train_pred = pred_prob(model,X_train)
y_test_pred = pred_prob(model,X_test)

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.close
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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [191]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give
 the least fpr
best_t = find_best_threshold(tr_thresholds, tra
in_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_be
st_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.667760
0002660056 for threshold 0.828
Train confusion matrix
[[ 2434   661]
 [ 2566 14439]]
```

In [192]:

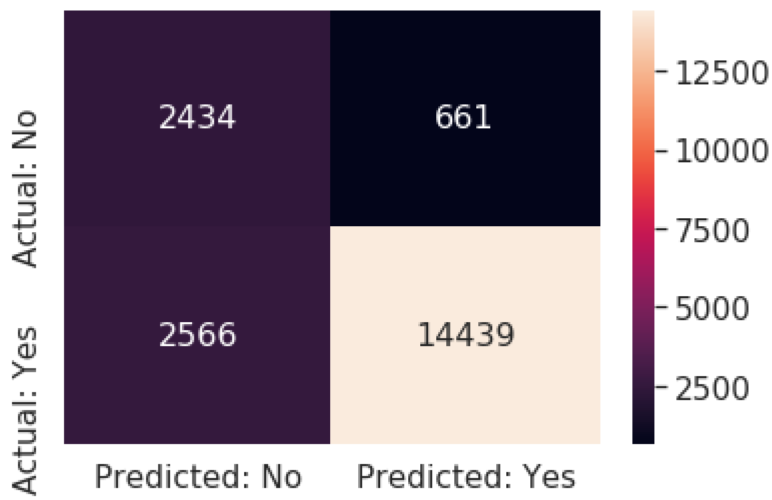
```
#plotting confusion matrix using seaborn's heat
map
# https://stackoverflow.com/questions/35572000/
how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confus
ion_matrix(y_train, predict_with_best_t(y_train
```

```
_pred, best_t)), ['Actual: No','Actual: Yes'],[
'Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=Tr
ue,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[192]:

```
<matplotlib.axes._subplots.AxesSubplot at
0x7f7d68747240>
```



In [193]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_bes
t_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 952  573]
 [2537 5838]]
```

In [194]:

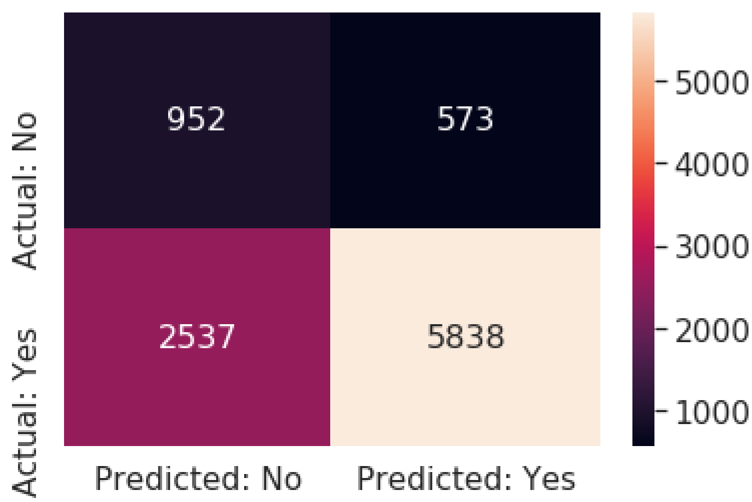
```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusi
on_matrix(y_test, predict_with_best_t(y_test_pr
```

```
ed, best_t)), ['Actual: No','Actual: Yes'],['Pr
edicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=Tru
e,annot_kws={"size": 16}, fmt='g')
```

Test data confusion matrix

Out[194]:

```
<matplotlib.axes._subplots.AxesSubplot at
0x7f7d687534e0>
```



2.5 Applying GBDT

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

2.5.1 Applying XGBOOST on BOW, SET 1

In [0]:

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

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

X_train = hstack((cat_0_train_normalized, cat_1_train_normalized, subcat_0_train_normalized, subcat_1_train_normalized, state_0_train_normalized, state_1_train_normalized, grade_0_train_normalized, grade_1_train_normalized, prefix_0_train_normalized, prefix_1_train_normalized, price_normalized_train, quantity_normalized_train, previously_posted_projects_normalized_train, title_word_count_normalized_train, essay_word_count_normalized_train, sent_pos_train, sent_neg_train, sent_neu_train, sent_compound_train, train_title_bow, train_essay_bow)).tocsr()
X_test =  hstack((cat_0_test_normalized, cat_1_test_normalized, subcat_0_test_normalized, subcat_1_test_normalized, state_0_test_normalized, state_1_test_normalized, grade_0_test_normalized, grade_1_test_normalized, prefix_0_test_normalized, prefix_1_test_normalized, price_normalized_test, quantity_normalized_test, previously_posted_projects_normalized_test, title_word_count_normalized_test, essay_word_count_normalized_test, sent_pos_test, sent_neg_test, sent_neu_test, sent_compound_test, test_title_bow, test_essay_bow)).tocsr()
```

In [196]:

```
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV
from xgboost import XGBClassifier
```

```
gbdt = XGBClassifier()

grid_params = {'n_estimators': [5, 10, 15, 20, 25, 30, 35], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 10]}

rs = RandomizedSearchCV(gbdt,grid_params ,cv=3, scoring='roc_auc',n_jobs=-1,return_train_score=True)
rs.fit(X_train, y_train)
```

Out[196]:

```
RandomizedSearchCV(cv=3, error_score='raise-deprecating',
                   estimator=XGBClassifier(base_score=0.5, booster='gbtree',

colsample_bylevel=1,

colsample_bynode=1,

colsample_bytree=1, gamma=0,

learning_rate=0.1, max_delta_step=0,

max_depth=3, min_child_weight=1,

missing=None, n_estimators=100,

n_jobs=1, nthread=None,

objective='binary:logistic',

random_state=0, reg_alpha=0,

reg_lambda=1, scale_pos_weight=1,
```

```
seed=None, silent=None, subsample=1,

verbosity=1),
                    iid='warn', n_iter=10,
n_jobs=-1,
                    param_distributions=
{'max_depth': [2, 3, 4, 5, 6, 7, 8, 9,

10],

'n_estimators': [5, 10, 15, 20, 25, 30,

35]},
                    pre_dispatch='2*n_job
s', random_state=None, refit=True,
                    return_train_score=Tru
e, scoring='roc_auc', verbose=0)
```

In [197]:

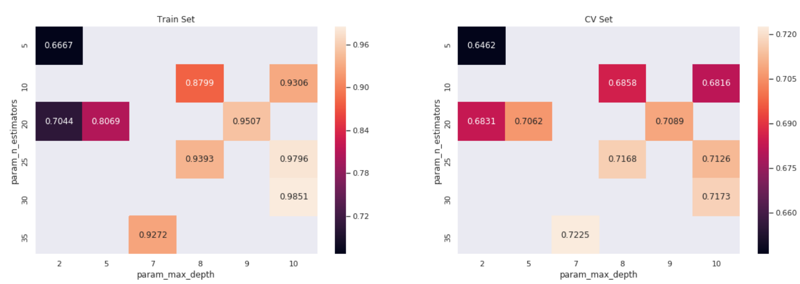
```
print('Best score: ',rs.best_score_)
print('k value with best score: ',rs.best_param
s_)
print('='*75)
print('Train AUC scores')
print(rs.cv_results_['mean_train_score'])
print('CV AUC scores')
print(rs.cv_results_['mean_test_score'])
```

```
Best score:  0.7224656227341516
k value with best score:  {'n_estimator
s': 35, 'max_depth': 7}
=========================================
==================================

Train AUC scores
[0.66671989 0.87993254 0.80686491 0.93056
107 0.97964675 0.70435178
```

```
 0.93931421 0.92721036 0.9850908  0.95073
755]
CV AUC scores
[0.64617362 0.68580412 0.70622284 0.68162
801 0.7125986  0.68314819
 0.71675846 0.72246562 0.71728043 0.70894
379]
```

In [198]:

```python
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(rs.cv_results_).grou
pby(['param_n_estimators', 'param_max_depth']).
max().unstack()[['mean_test_score', 'mean_train
_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot
= True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot
= True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



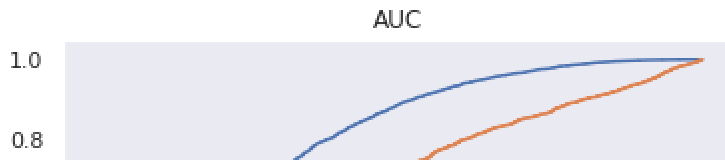
In [0]:

```python
max_d = rs.best_params_['max_depth']
n_est = rs.best_params_['n_estimators']
```

In [200]:

```python
# https://scikit-learn.org/stable/modules/gener
ated/sklearn.metrics.roc_curve.html#sklearn.met
rics.roc_curve
```

```
from sklearn.metrics import roc_curve, auc
model = RandomForestClassifier(max_depth = max_
d, n_estimators = n_est)

model.fit(X_train,y_train)

y_train_pred = pred_prob(model,X_train)
y_test_pred = pred_prob(model,X_test)

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.close
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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [201]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
```

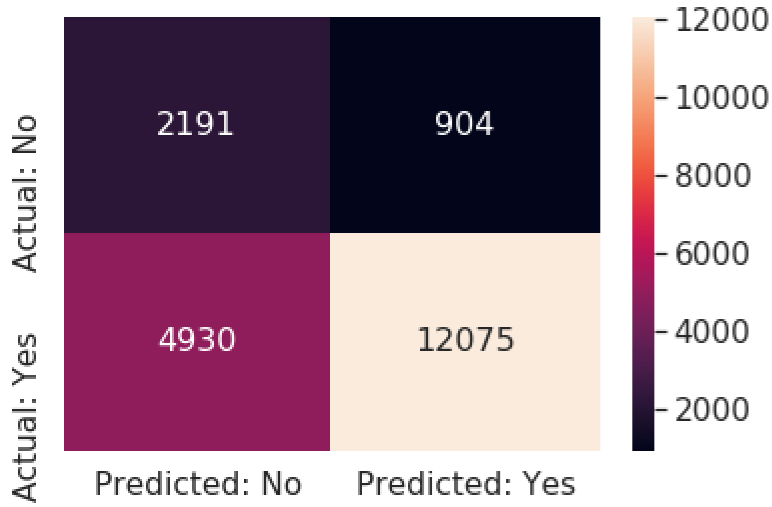
```
the maximum value of tpr*(1-fpr) 0.5026807187280753 for threshold 0.842
Train confusion matrix
[[ 2191   904]
 [ 4930 12075]]
```

In [202]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[202]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7d68698320>



In [203]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 672  853]
 [1978 6397]]
```

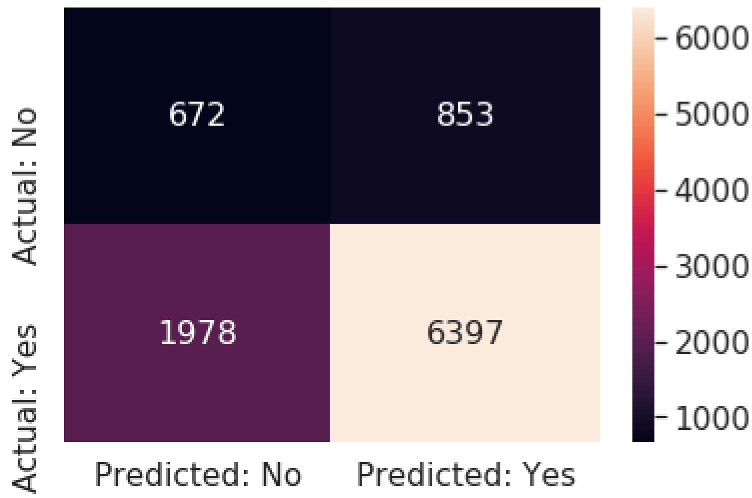
In [204]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[204]: <matplotlib.axes._subplots.AxesSubplot at

```
<matplotlib.axes._subplots.AxesSubplot at
0x7f7d67d87128>
```



2.5.2 Applying XGBOOST on TFIDF, SET 2

In [0]:

```
# Please write all the code with proper documentation
# Please write all the code with proper documentation
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_train = hstack((cat_0_train_normalized, cat_1_train_normalized, subcat_0_train_normalized, subcat_1_train_normalized, state_0_train_normalized, state_1_train_normalized, grade_0_train_normalized, grade_1_train_normalized, prefix_0_train_normalized, prefix_1_train_normalized, price_normalized_train, quantity_normalized_train, previously_posted_projects_normalized_train, title_word_count_normalized_train, essay_word_cou
```

```
nt_normalized_train, sent_pos_train, sent_neg_train, sent_neu_train, sent_compound_train, train_title_tfidf, train_essay_tfidf)).tocsr()
X_test =  hstack((cat_0_test_normalized, cat_1_test_normalized, subcat_0_test_normalized, subcat_1_test_normalized, state_0_test_normalized, state_1_test_normalized, grade_0_test_normalized, grade_1_test_normalized, prefix_0_test_normalized, prefix_1_test_normalized, price_normalized_test, quantity_normalized_test, previously_posted_projects_normalized_test, title_word_count_normalized_test, essay_word_count_normalized_test, sent_pos_test, sent_neg_test, sent_neu_test, sent_compound_test, test_title_tfidf, test_essay_tfidf)).tocsr()
```

In [206]:

```
from scipy.stats import randint as sp_randint
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier

gbdt = XGBClassifier()

grid_params = {'n_estimators': [5, 10, 15, 20, 25, 30, 35], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 10]}

gs = GridSearchCV(gbdt,grid_params ,cv=3, scoring='roc_auc',n_jobs=-1,return_train_score=True)
gs.fit(X_train, y_train)
```

Out[206]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
             estimator=XGBClassifier(base_score=0.5, booster='gbtree',
```

```
                                                      cols
ample_bylevel=1, colsample_bynode=1,
                                                      cols
ample_bytree=1, gamma=0,
                                                      lear
ning_rate=0.1, max_delta_step=0,
                                                      max_
depth=3, min_child_weight=1,
                                                      miss
ing=None, n_estimators=100, n_jobs=1,
                                                      nthr
ead=None, objective='binary:logistic',
                                                      rand
om_state=0, reg_alpha=0, reg_lambda=1,
                                                      scal
e_pos_weight=1, seed=None, silent=None,
                                                      subs
ample=1, verbosity=1),
             iid='warn', n_jobs=-1,
             param_grid={'max_depth': [2,
3, 4, 5, 6, 7, 8, 9, 10],
                         'n_estimators':
[5, 10, 15, 20, 25, 30, 35]},
             pre_dispatch='2*n_jobs', ref
it=True, return_train_score=True,
             scoring='roc_auc', verbose=
0)
```

In [207]:

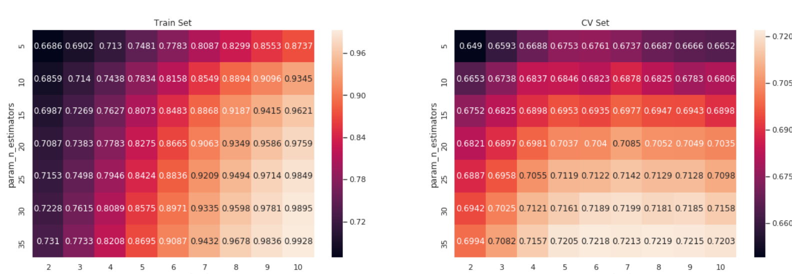
```
print('Best score: ',gs.best_score_)
print('k value with best score: ',gs.best_param
s_)
print('='*75)
print('Train AUC scores')
print(gs.cv_results_['mean_train_score'])
print('CV AUC scores')
print(gs.cv_results_['mean_test_score'])
```

```
Best score:  0.7218742581592476
k value with best score:  {'max_depth':
8, 'n_estimators': 35}
==========================================
==================================
Train AUC scores
[0.66857266 0.68585384 0.69874443 0.70866
975 0.71526946 0.72282196
 0.73104384 0.69024092 0.7140186  0.72689
318 0.73832239 0.74981227
 0.76147949 0.77334283 0.71302453 0.74378
841 0.76265276 0.77833098
 0.79461603 0.80887957 0.82081784 0.74806
206 0.78341998 0.80727719
 0.82749786 0.8424464  0.8575041  0.86952
897 0.77828171 0.81581703
 0.84826641 0.86652505 0.88363047 0.89714
301 0.90874264 0.8086753
 0.85485564 0.8868493  0.90632622 0.92085
253 0.93348911 0.94320957
 0.8298689  0.88939852 0.9186513  0.93487
138 0.94941867 0.95984765
 0.96783049 0.85534691 0.90955808 0.94150
98  0.95855569 0.97135203
 0.97810092 0.98362833 0.87367278 0.93452
01  0.96209464 0.97590501
 0.98485613 0.9895416  0.9927766 ]
CV AUC scores
[0.64897532 0.66532694 0.67521816 0.68208
349 0.68870916 0.69419
 0.69938642 0.6593254  0.67380733 0.68251
65  0.68966361 0.69580853

 0.70249844 0.70819091 0.66878915 0.68373
908 0.68978693 0.6980769
 0.70548975 0.71209832 0.71574862 0.67527
```

```
967 0.68463274 0.69525573
 0.70366069 0.71187975 0.71613983 0.72045
979 0.67611194 0.68226276
 0.69350742 0.70403044 0.71215577 0.71893
866 0.72178394 0.67367887
 0.6877852  0.69766889 0.70848815 0.71421
35  0.71987314 0.72128027
 0.66868997 0.68246366 0.69468661 0.70520
048 0.71287483 0.71812359
 0.72187426 0.66661664 0.67825753 0.69427
5   0.70490979 0.71276794
 0.71847356 0.72151727 0.6651854  0.68055
311 0.68980557 0.70351528
 0.70983713 0.71577252 0.72028475]
```

In [208]:

```
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(gs.cv_results_).grou
pby(['param_n_estimators', 'param_max_depth']).
max().unstack()[['mean_test_score', 'mean_train
_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot
= True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot
= True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```

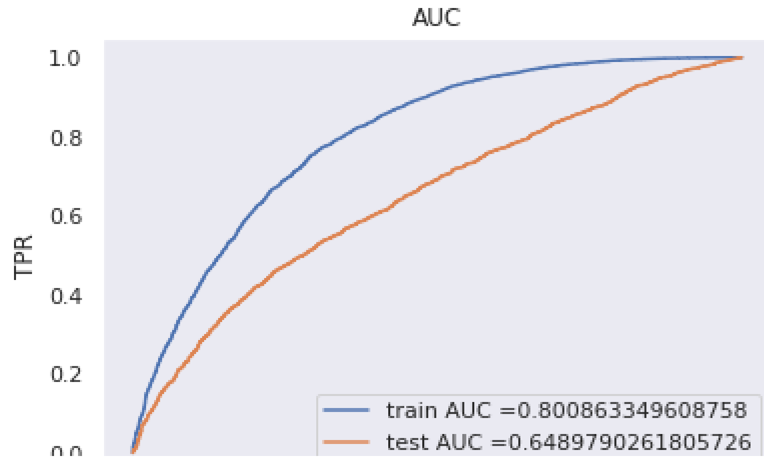


In [0]:

```
max_d = rs.best_params_['max_depth']
n_est = rs.best_params_['n_estimators']
```

In [210]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
model = RandomForestClassifier(max_depth = max_d, n_estimators = n_est)

model.fit(X_train,y_train)

y_train_pred = pred_prob(model,X_train)
y_test_pred = pred_prob(model,X_test)

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.close
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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
plt.grid()
plt.show()
```



In [211]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give
 the least fpr
best_t = find_best_threshold(tr_thresholds, tra
in_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_be
st_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.532421
3205371982 for threshold 0.843
Train confusion matrix
[[ 2157   938]
 [ 4014 12991]]
```

In [212]:

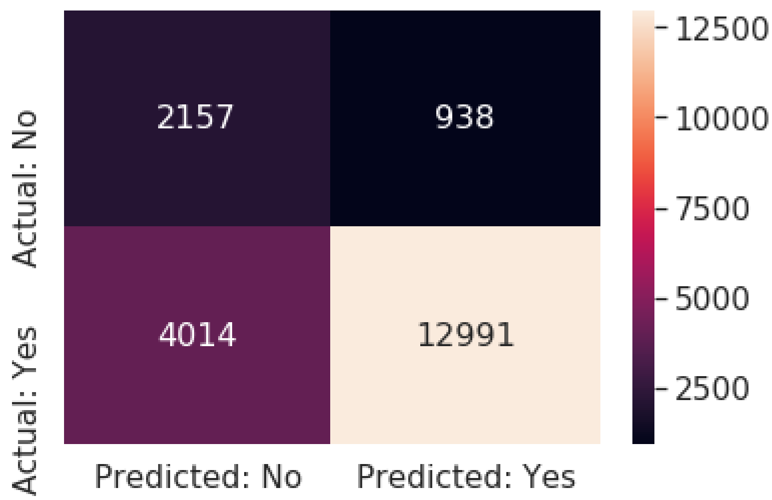
```
#plotting confusion matrix using seaborn's heat
map
# https://stackoverflow.com/questions/35572000/
how-can-i-plot-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confus
ion_matrix(y_train, predict_with_best_t(y_train
```

```
_pred, best_t)), ['Actual: No','Actual: Yes'],[
'Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[212]:

```
<matplotlib.axes._subplots.AxesSubplot at
0x7f7d67d58080>
```



In [213]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 841  684]
 [2932 5443]]
```

In [214]:

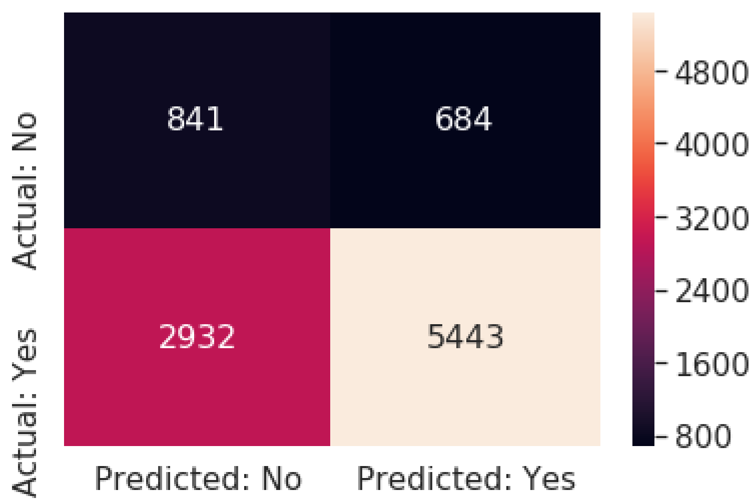
```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pr
```

```
ed, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[214]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d6798a240>
```



2.5.3 Applying XGBOOST on AVG W2V, SET 3

In [0]:

```
# Please write all the code with proper documentation
train_avg_w2v_essays_np = np.array(train_avg_w2v_essays)
train_avg_w2v_titles_np = np.array(train_avg_w2v_titles)
test_avg_w2v_essays_np  = np.array(test_avg_w2v_essays)
```

```
test_avg_w2v_titles_np  = np.array(test_avg_w2v
_titles)
```

In [0]:

```
#https://blog.csdn.net/w55100/article/details/9
0369779
# if you use hstack without converting it into
 to a sparse matrix first,
#it shows an error: blocks must be 2-D

from scipy.sparse import coo_matrix, hstack
tr1 = coo_matrix(cat_0_train_normalized)
tr2 = coo_matrix(cat_1_train_normalized)
tr3 = coo_matrix(subcat_0_train_normalized)
tr4 = coo_matrix(subcat_1_train_normalized)
tr5 = coo_matrix(state_0_train_normalized)
tr6 = coo_matrix(state_1_train_normalized)
tr7 = coo_matrix(grade_0_train_normalized)
tr8 = coo_matrix(grade_1_train_normalized)
tr9 = coo_matrix(prefix_0_train_normalized)
tr10 = coo_matrix(prefix_1_train_normalized)
tr11 = coo_matrix(price_normalized_train)
tr12 = coo_matrix(quantity_normalized_train)
tr13 = coo_matrix(previously_posted_projects_no
rmalized_train)
tr14 = coo_matrix(title_word_count_normalized_t
rain)
tr15 = coo_matrix(essay_word_count_normalized_t
rain)
tr16 = coo_matrix(sent_pos_train)
tr17 = coo_matrix(sent_neg_train)
tr18 = coo_matrix(sent_neu_train)
tr19 = coo_matrix(sent_compound_train)
tr20 = coo_matrix(train_avg_w2v_essays_np)
tr21 = coo_matrix(train_avg_w2v_titles_np)
```

In [0]:

```
X_train = hstack([tr1,tr2,tr3,tr4,tr5,tr6,tr7,t
r8,tr9,tr10,tr11,tr12,tr13,tr14,tr15,tr16,tr17,
tr18,tr19,tr20,tr21]).tocsr()
```

In [0]:

```
te1 = coo_matrix(cat_0_test_normalized)
te2 = coo_matrix(cat_1_test_normalized)
te3 = coo_matrix(subcat_0_test_normalized)
te4 = coo_matrix(subcat_1_test_normalized)
te5 = coo_matrix(state_0_test_normalized)
te6 = coo_matrix(state_1_test_normalized)
te7 = coo_matrix(grade_0_test_normalized)
te8 = coo_matrix(grade_1_test_normalized)
te9 = coo_matrix(prefix_0_test_normalized)
te10 = coo_matrix(prefix_1_test_normalized)
te11 = coo_matrix(price_normalized_test)
te12 = coo_matrix(quantity_normalized_test)
te13 = coo_matrix(previously_posted_projects_no
rmalized_test)
te14 = coo_matrix(title_word_count_normalized_t
est)
te15 = coo_matrix(essay_word_count_normalized_t
est)
te16 = coo_matrix(sent_pos_test)
te17 = coo_matrix(sent_neg_test)
te18 = coo_matrix(sent_neu_test)
te19 = coo_matrix(sent_compound_test)
te20 = coo_matrix(test_avg_w2v_essays_np)
te21 = coo_matrix(test_avg_w2v_titles_np)
```

In [0]:

```
X_test = hstack([te1,te2,te3,te4,te5,te6,te7,te
8,te9,te10,te11,te12,te13,te14,te15,te16,te17,t
e18,te19,te20,te21]).tocsr()
```

In [220]:

```
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedS
```

```
earchCV
from xgboost import XGBClassifier

gbdt = XGBClassifier()

grid_params = {'n_estimators': [5, 10, 15, 20,
25, 30, 35], 'max_depth':[2, 3, 4, 5, 6, 7, 8,
9, 10]}

rs = RandomizedSearchCV(gbdt,grid_params ,cv=3,
scoring='roc_auc',n_jobs=-1,return_train_score=
True)
rs.fit(X_train, y_train)
```

Out[220]:

```
RandomizedSearchCV(cv=3, error_score='rai
se-deprecating',
                   estimator=XGBClassifie
r(base_score=0.5, booster='gbtree',

colsample_bylevel=1,

colsample_bynode=1,

colsample_bytree=1, gamma=0,

learning_rate=0.1, max_delta_step=0,

max_depth=3, min_child_weight=1,

missing=None, n_estimators=100,

n_jobs=1, nthread=None,

objective='binary:logistic',

random_state=0, reg_alpha=0,
```

```
reg_lambda=1, scale_pos_weight=1,

seed=None, silent=None, subsample=1,

verbosity=1),
                   iid='warn', n_iter=10,
n_jobs=-1,
                   param_distributions=
{'max_depth': [2, 3, 4, 5, 6, 7, 8, 9,

10],

'n_estimators': [5, 10, 15, 20, 25, 30,

35]},
                   pre_dispatch='2*n_job
s', random_state=None, refit=True,
                   return_train_score=Tru
e, scoring='roc_auc', verbose=0)
```

In [221]:

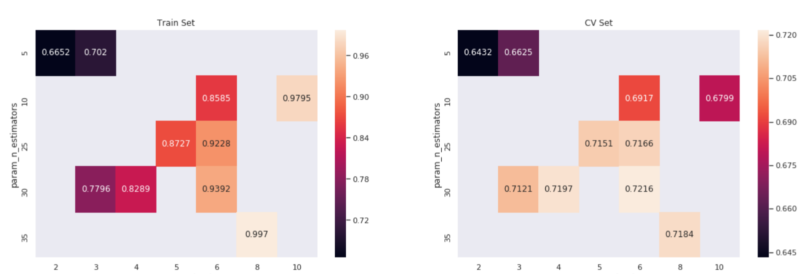
```
print('Best score: ',rs.best_score_)
print('k value with best score: ',rs.best_param
s_)
print('='*75)
print('Train AUC scores')
print(rs.cv_results_['mean_train_score'])
print('CV AUC scores')
print(rs.cv_results_['mean_test_score'])
```

```
Best score:  0.7215894718873592
k value with best score:  {'n_estimator
s': 30, 'max_depth': 6}
=========================================
==================================
```

```
Train AUC scores
[0.87266231 0.99697    0.70198026 0.77964604 0.92282842 0.85849516
 0.93921288 0.66522938 0.82894018 0.97954267]
CV AUC scores
[0.71510919 0.71837512 0.66254084 0.71207049 0.71659669 0.69170919
 0.72158947 0.64316483 0.71966987 0.6799085 ]
```

In [222]:

```python
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(rs.cv_results_).groupby(['param_n_estimators', 'param_max_depth']).max().unstack()[['mean_test_score', 'mean_train_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



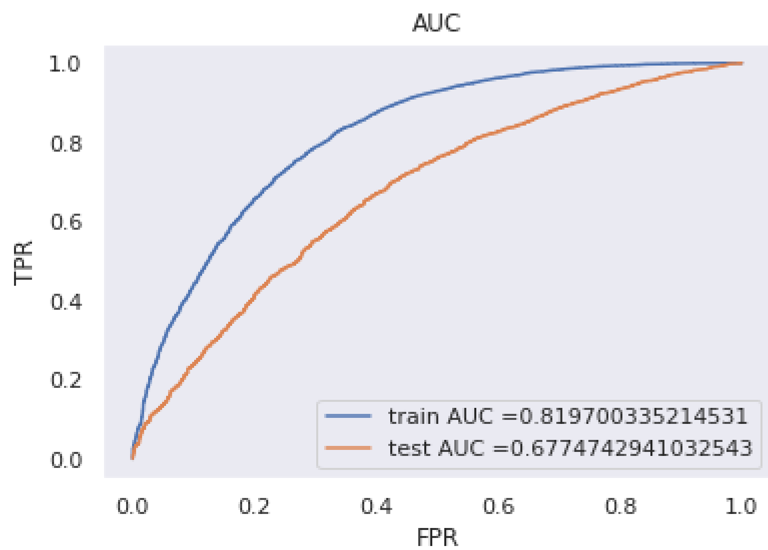
In [223]:
```
rs.best_params_
```

Out[223]:
```
{'max_depth': 6, 'n_estimators': 30}
```

In [0]:
```
max_d = rs.best_params_['max_depth']
n_est = rs.best_params_['n_estimators']
```

In [225]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
model = RandomForestClassifier(max_depth = max_d, n_estimators = n_est)

model.fit(X_train,y_train)

y_train_pred = pred_prob(model,X_train)
y_test_pred = pred_prob(model,X_test)

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.close
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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
```

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



In [226]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give
 the least fpr
best_t = find_best_threshold(tr_thresholds, tra
in_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_be
st_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.552180
7659915666 for threshold 0.836
Train confusion matrix
[[ 2192   903]
 [ 3747 13258]]
```

In [227]:

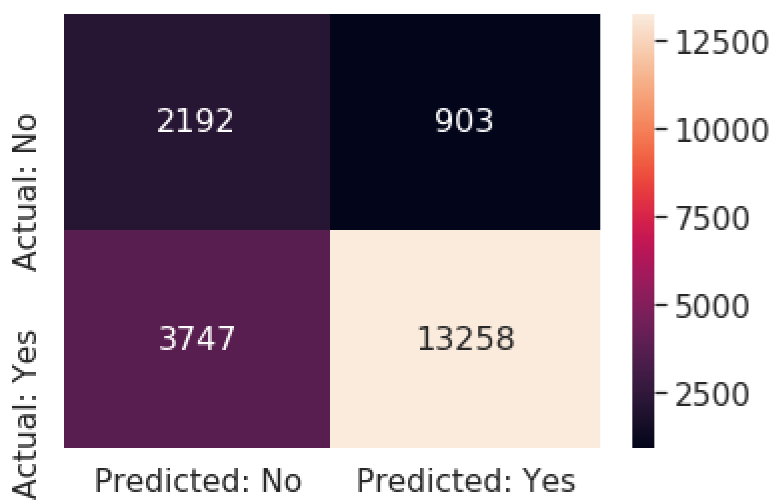
```
#plotting confusion matrix using seaborn's heat
map
# https://stackoverflow.com/questions/35572000/
how-can-i-plot-a-confusion-matrix
```

```
print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[227]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d67a3e400>
```



In [228]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

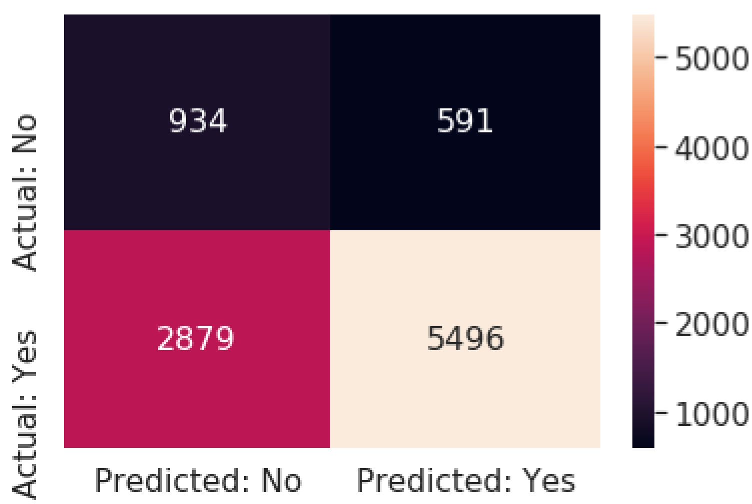
```
Test confusion matrix
[[ 934  591]
 [2879 5496]]
```

In [229]:
```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

Test data confusion matrix

Out[229]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d67aa1c88>
```



2.5.4 Applying XGBOOST on TFIDF W2V, SET 4

In [0]:
```
# Please write all the code with proper documentation
train_tfidf_w2v_essays_np = np.array(train_tfidf_w2v_essays)
```

```
train_tfidf_w2v_titles_np = np.array(train_tfid
f_w2v_titles)
test_tfidf_w2v_essays_np  = np.array(test_tfidf
_w2v_essays)
test_tfidf_w2v_titles_np  = np.array(test_tfidf
_w2v_titles)
```

In [0]:

```
#https://blog.csdn.net/w55100/article/details/9
0369779
# if you use hstack without converting it into
 to a sparse matrix first,
#it shows an error: blocks must be 2-D

from scipy.sparse import coo_matrix, hstack
tr1 = coo_matrix(cat_0_train_normalized)
tr2 = coo_matrix(cat_1_train_normalized)
tr3 = coo_matrix(subcat_0_train_normalized)
tr4 = coo_matrix(subcat_1_train_normalized)
tr5 = coo_matrix(state_0_train_normalized)
tr6 = coo_matrix(state_1_train_normalized)
tr7 = coo_matrix(grade_0_train_normalized)
tr8 = coo_matrix(grade_1_train_normalized)
tr9 = coo_matrix(prefix_0_train_normalized)
tr10 = coo_matrix(prefix_1_train_normalized)
tr11 = coo_matrix(price_normalized_train)
tr12 = coo_matrix(quantity_normalized_train)
tr13 = coo_matrix(previously_posted_projects_no
rmalized_train)
tr14 = coo_matrix(title_word_count_normalized_t
rain)
tr15 = coo_matrix(essay_word_count_normalized_t
rain)
tr16 = coo_matrix(sent_pos_train)
tr17 = coo_matrix(sent_neg_train)
tr18 = coo_matrix(sent_neu_train)
```

```
tr19 = coo_matrix(sent_compound_train)
tr20 = coo_matrix(train_tfidf_w2v_essays_np)
tr21 = coo_matrix(train_tfidf_w2v_titles_np)
```

In [0]:
```
X_train = hstack([tr1,tr2,tr3,tr4,tr5,tr6,tr7,tr8,tr9,tr10,tr11,tr12,tr13,tr14,tr15,tr16,tr17,tr18,tr19,tr20,tr21]).tocsr()
```

In [0]:
```
te1 = coo_matrix(cat_0_test_normalized)
te2 = coo_matrix(cat_1_test_normalized)
te3 = coo_matrix(subcat_0_test_normalized)
te4 = coo_matrix(subcat_1_test_normalized)
te5 = coo_matrix(state_0_test_normalized)
te6 = coo_matrix(state_1_test_normalized)
te7 = coo_matrix(grade_0_test_normalized)
te8 = coo_matrix(grade_1_test_normalized)
te9 = coo_matrix(prefix_0_test_normalized)
te10 = coo_matrix(prefix_1_test_normalized)
te11 = coo_matrix(price_normalized_test)
te12 = coo_matrix(quantity_normalized_test)
te13 = coo_matrix(previously_posted_projects_normalized_test)
te14 = coo_matrix(title_word_count_normalized_test)
te15 = coo_matrix(essay_word_count_normalized_test)
te16 = coo_matrix(sent_pos_test)
te17 = coo_matrix(sent_neg_test)
te18 = coo_matrix(sent_neu_test)
te19 = coo_matrix(sent_compound_test)
te20 = coo_matrix(test_tfidf_w2v_essays_np)
te21 = coo_matrix(test_tfidf_w2v_titles_np)
```

In [0]:
```
X_test = hstack([te1,te2,te3,te4,te5,te6,te7,te8,te9,te10,te11,te12,te13,te14,te15,te16,te17,t
```

```
e18,te19,te20,te21]).tocsr()
```

In [235]:

```
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV
from xgboost import XGBClassifier

gbdt = XGBClassifier()

grid_params = {'n_estimators': [5, 10, 15, 20, 25, 30, 35], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 10]}

rs = RandomizedSearchCV(gbdt,grid_params ,cv=3, scoring='roc_auc',n_jobs=-1,return_train_score=True)
rs.fit(X_train, y_train)
```

Out[235]:

```
RandomizedSearchCV(cv=3, error_score='raise-deprecating',
                   estimator=XGBClassifier(base_score=0.5, booster='gbtree',
                                           colsample_bylevel=1,
                                           colsample_bynode=1,
                                           colsample_bytree=1, gamma=0,
                                           learning_rate=0.1, max_delta_step=0,
                                           max_depth=3, min_child_weight=1,
                                           missing=None, n_estimators=100,
                                           n_jobs=1, nthread=None,
```

```
n_jobs=1, nthread=None,

objective='binary:logistic',

random_state=0, reg_alpha=0,

reg_lambda=1, scale_pos_weight=1,

seed=None, silent=None, subsample=1,

verbosity=1),
                   iid='warn', n_iter=10,
n_jobs=-1,
                   param_distributions=
{'max_depth': [2, 3, 4, 5, 6, 7, 8, 9,

10],

'n_estimators': [5, 10, 15, 20, 25, 30,

35]},
                   pre_dispatch='2*n_job
s', random_state=None, refit=True,
                   return_train_score=Tru
e, scoring='roc_auc', verbose=0)
```

In [236]:

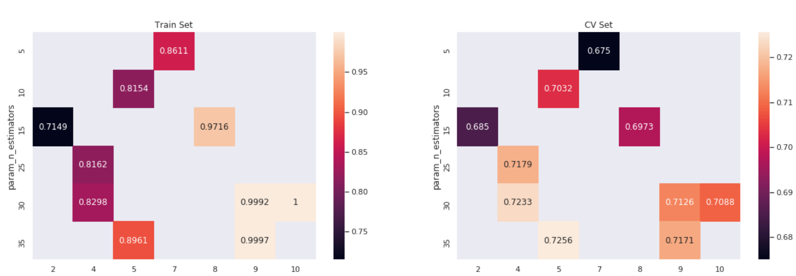
```
print('Best score: ',rs.best_score_)
print('k value with best score: ',rs.best_param
s_)
print('='*75)
print('Train AUC scores')
print(rs.cv_results_['mean_train_score'])
print('CV AUC scores')
print(rs.cv_results_['mean_test_score'])
```

```
Best score:  0.725589730388709
```

```
k value with best score:  {'n_estimator
s': 35, 'max_depth': 5}
=========================================
==================================
Train AUC scores
[0.99966909 0.71488734 0.81624348 0.89613
305 0.82975511 0.86114728
 0.99921299 0.99997944 0.81542896 0.97164
079]
CV AUC scores
[0.7170925  0.68497315 0.71791127 0.72558
973 0.72326732 0.67503654
 0.71259468 0.7088266  0.70315761 0.69727
265]
```

In [237]:

```python
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(rs.cv_results_).grou
pby(['param_n_estimators', 'param_max_depth']).
max().unstack()[['mean_test_score', 'mean_train
_score']]
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot
= True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot
= True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [238]:

```
rs.best_params_
```

Out[238]:

```
{'max_depth': 5, 'n_estimators': 35}
```

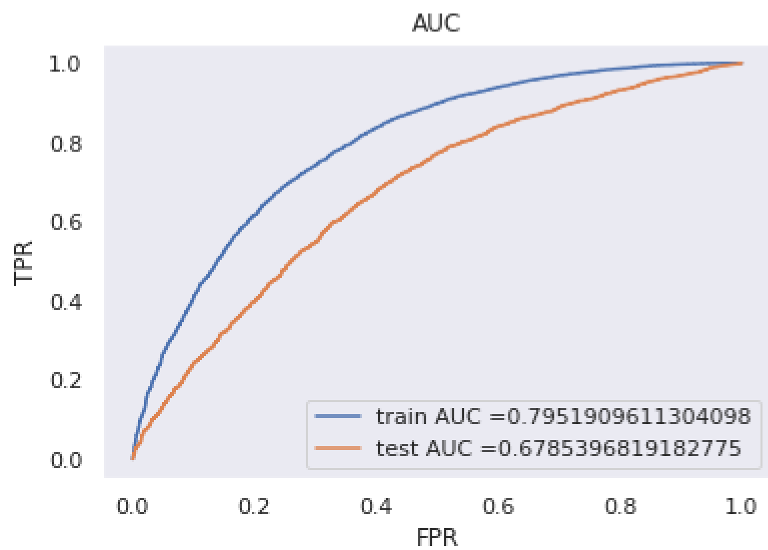
In [0]:

```
max_d = rs.best_params_['max_depth']
n_est = rs.best_params_['n_estimators']
```

In [240]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
model = RandomForestClassifier(max_depth = max_d, n_estimators = n_est)

model.fit(X_train,y_train)

y_train_pred = pred_prob(model,X_train)
y_test_pred = pred_prob(model,X_test)

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.close
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("FPR")
plt.ylabel("TPR")
plt.title("AUC")
```

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



In [241]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give
 the least fpr
best_t = find_best_threshold(tr_thresholds, tra
in_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_be
st_t(y_train_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.521310
2294820634 for threshold 0.838
Train confusion matrix
[[ 2237   858]
 [ 4740 12265]]
```

In [242]:

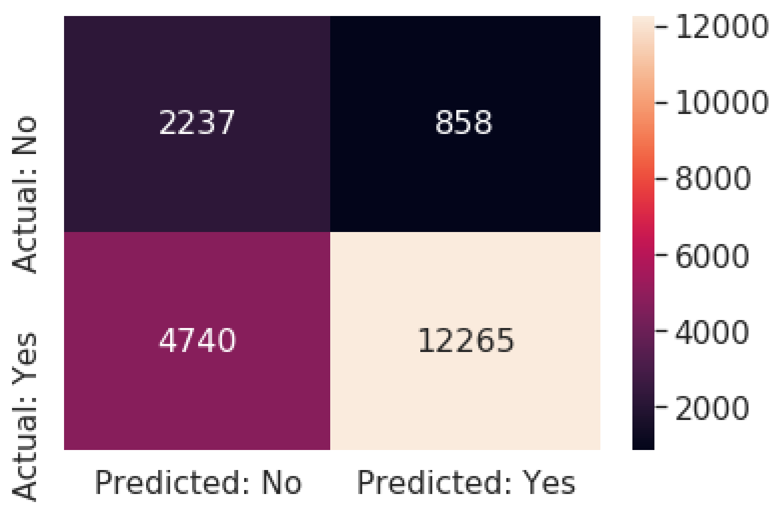
```
#plotting confusion matrix using seaborn's heat
map
# https://stackoverflow.com/questions/35572000/
how-can-i-plot-a-confusion-matrix
```

```
print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[242]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d67aa8630>
```



In [243]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
Test confusion matrix
[[ 954  571]
 [2952 5423]]
```

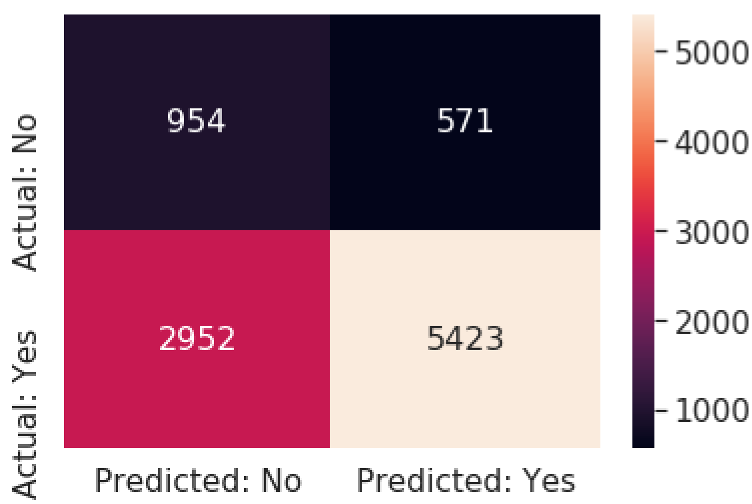
In [244]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)), ['Actual: No','Actual: Yes'],['Predicted: No','Predicted: Yes'])
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

Test data confusion matrix

Out[244]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7d688b50b8>
```



3. Conclusion

In [246]:

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

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

```
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

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

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyperp
arameters(n_estimators,max_depth)", "Test AUC"]

x.add_row(["BOW", "RF","(1000, 9)", 0.7141])
x.add_row(["TFIDF", "RF", "(1000, 9)",  0.7105
])
x.add_row(["AVG W2V", "RF", "(300, 7)",  0.7080
])
x.add_row(["TFIDF W2V", "RF", "(200, 7)",  0.70
92])

x.add_row(["----- ---", "----", "--------------
-----------------", "---------"])

x.add_row(["BOW", "GBDT","(35, 7)",  0.6553])
x.add_row(["TFIDF", "GBDT", "(35, 8)",  0.6489
])
x.add_row(["AVG W2V", "GBDT", "(30, 6)",  0.677
4])
x.add_row(["TFIDF W2V", "GBDT", "(35, 5)", 0.67
53])



print(x)
```

```
+------------+-------+-----------------
```