Document Created Date:04-06-2019

DonorsChoose.org
Support a classroom. Build a future.

Understanding data and data source - DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	
`project_id`	A unique identifier for the proposed project. **Exampl**
`project_title`	Title of the projec • `Art Will Make Y` • `First`

Feature	
`project_grade_category`	Grade level of students for which the project is targeted. One of t enumer • `Grad` • `G` • `G` • `Gr`
`project_subject_categories`	One or more (comma-separated) subject categories for the pro following enumerated li • `Applied` • `Care` • `Health` • `History` • `Literacy &` • `Math` • `Music &` • `Spec` • • `Music &` • `Literacy & Language, Math`
`school_state`	State where school is located ([Two-letter U.S. (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos) **Ex**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories fo • • `Literature & Writing, Social`
`project_resource_summary`	An explanation of the resources needed for the projec • `My students need hands on literacy materials t` `sensory nee`
`project_essay_1`	First applic
`project_essay_2`	Second applic
`project_essay_3`	Third applic
`project_essay_4`	Fourth applic
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `20` `12:`
`teacher_id`	A unique identifier for the teacher of the proposed proje `bdf8baa8fedef6bfeec7ae4`
`teacher_prefix`	Teacher's title. One of the following enumera • • • • • •
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the sa **I**

* 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.

Random Forest and XGBoost

`Step by Step Procedure`

- Understanding the Businessreal world problem
- Loading the data
- Preprocessing the data(based on the type of data = categorical , text, Numarical)
- Preprocessing data includes (removing outliers, impute missung values, cleaning data,etc..)
- Split the data into train, cv, test
- Vectorization data (one hot encoding)

- Vectorizing text data
- Normalizing
- Contactinating all the type of features(cat + text + num)
- Hyperparameter tuning to find th best estimator(GridSearch)
- Ploting the performence of the model using heatmaps
- Train the Random Forest model using best hyperparameter and ploting auc roc-curve
- Plot confusion matrix
- Hyperparameter tuning to find th best estimator(RandomizedSearch)
- Ploting the performence of the model using heatmaps
- Train the XGBoost model using best hyperparameter and ploting auc roc-curve
- Plot Confusion Matrix
- Observation on overall model performences
- Ploting the performences by table format.

```
C:\Users\Ramesh Battu> import required libraries
```

In [1]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os
```

1.1 Reading Data

In [2]:

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

In [3]:

```
print("Number of data points in train data", project_data.shape)
print('-'*88)
print("The attributes of data :", project_data.columns.values)
print('-'*88)
```

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

In [4]:

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

#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40840
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)

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

project_data.head(2)
```

Out[4]:

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

In [5]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
print('-'*60)
resource_data.head(2)
```

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

Out[5]:

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

1.1.1 Preprocessing of project_subject_categories

In [6]:

```
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
catogories = list(project_data['project_subject_categories'].values)

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

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

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

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

1.1.2 Preprocessing of project_subject_subcategories

In [7]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth

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

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

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

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

1.1.3 Preprocessing of `school_state`

In [8]:

```
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
school_state_catogories = list(project_data['school_state'].values)
cat_list = []
for i in school_state_catogories:
    temp = "" # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
project_data['school_state'] = cat_list

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

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

1.1.4 Preprocessing of `teacher_prefix`

In [9]:

```
# citation code :https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['teacher_prefix'].value_counts().index[
teacher_prefix_catogories = list(project_data['teacher_prefix'].values)
# Citation code : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-sciki
# To convert the data type object to unicode string : used """astype('U')""" code from
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
cat_list = []
for i in teacher_prefix_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            j=j.replace('The','') # if we have the words "The" we are going to replace 
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
        temp =temp.replace('.', '') # we are removing dot(.)
    cat_list.append(temp.strip())

project_data['teacher_prefix'] = cat_list

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
teacher_prefix_dict = dict(my_counter)
sorted_teacher_prefix_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda kv: kv
```

In [10]:

```
sorted_teacher_prefix_dict
```

Out[10]:

```
{'Dr': 13, 'Teacher': 2360, 'Mr': 10648, 'Ms': 38955, 'Mrs': 57272}
```

1.1.5 Preprocessing of `project_grade_category`

In [11]:

```
project_grade_catogories = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
cat_list = []
for i in project_grade_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
        temp = temp.replace('-','_') # we are replaceing '-' with '_'
    cat_list.append(temp.strip())

project_data['project_grade_category'] = cat_list

#link : https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['project_grade_category'].value_counts(

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_grade_category_dict = dict(my_counter)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items(), k
```

1.2. Text Preprocessing

1.2.1 Text Preprocessing of essay

In [12]:

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

In [13]:

```
project_data.head(1)
```

Out[13]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs	CA	00

In [14]:

```
# printing some random reviews
print(project_data['essay'].values[1])
print("="*125)
print(project_data['essay'].values[125])
print("="*125)
print(project_data['essay'].values[2020])
print("="*125)
print(project_data['essay'].values[40020])
print("="*125)
print(project_data['essay'].values[99999])
print("="*125)
```

```
Imagine being 8-9 years old. You're in your third grade classroom. You s
ee bright lights, the kid next to you is chewing gum, the birds are maki
ng noise, the street outside is buzzing with cars, it's hot, and your te
acher is asking you to focus on learning. Ack! You need a break! So do m
y students.Most of my students have autism, anxiety, another disability,
or all of the above. It is tough to focus in school due to sensory overl
oad or emotions. My students have a lot to deal with in school, but I th
ink that makes them the most incredible kids on the planet. They are kin
d, caring, and sympathetic. They know what it's like to be overwhelmed,
so they understand when someone else is struggling. They are open-minded
and compassionate. They are the kids who will someday change the world.I
t is tough to do more than one thing at a time. When sensory overload ge
ts in the way, it is the hardest thing in the world to focus on learnin
g. My students need many breaks throughout the day, and one of the best
items we've used is a Boogie Board. If we had a few in our own classroo
m, my students could take a break exactly when they need one, regardless
of which other rooms in the school are occupied. Many of my students nee
d to do something with their hands in order to focus on the task at han
d. Putty will give the sensory input they need in order to focus, it wil
l calm them when they are overloaded, it will help improve motor skills,
and it will make school more fun.When my students are able to calm thems
elves down, they are ready to learn. When they are able to focus, they w
ill learn more and retain more. They will get the sensory input they nee
d and it will prevent meltdowns (which are scary for everyone in the roo
m). This will lead to a better, happier classroom community that is able
to learn the most they can in the best way possible.
========================================================================
=====================================================
Seventh and eighth grade students at my school are getting to use the sc
hool's science lab for the first time this year. It is my hope that scie
nce will quickly become their favorite subject when they realize it is n
ot just a subject, but everything in the world around them.My students a
re the future leaders of their community. They are learning to set an ex
ample of excellence and service in all they do. Scholars in our middle s
chool program are working hard to gain the skills they need to succeed i
n today's competitive world, but are doing so in an environment that nur
tures individuals and encourages peace and thoughtfulness.Students will
each have a binder where they can keep assignments for their classes.  T
he Middle School Team will help the students learn to keep their work fo
r each class in a separate tab and incorporate a color-coded system. Kid
s will be able to personalize their binders and have them available at a
ll times to prevent assignments from getting lost.Adolescents are notori
ously forgetful and disorganized. Our team hopes that creating a fail-sa
fe organization plan, students will be able to keep track of their work
and important papers.
========================================================================
=====================================================
```

I have long dreamed of teaching Angels in America, a play for my AP students that stimulated their thoughts and understand the universal promise of the American dream.My students come from extremely diverse backgrounds.\r\n\r\nThere are students who have fled war-torn countries such as the Ukraine, to first generation Mexican-Americans, to students dealing with Asbergers and even a student who is in the advanced stages of Muscular Dystrophy. Through all their struggles, they are extremely resilient and come to school every day with hopes of a better future.In class we will be reading \"Angels in America\" together and discussing in Socratic seminars and writing papers on themes such as: visions of America, magical realism, the need for a sense of community in our lives, and how caustic and demeaning stereotyping can be.AP students are at an age in their lives where they are ready to see the world through many lenses. These unique perspectives make they not only better readers and writers, but also more prepared for the world they are entering.

==

A typical day in my classroom is filled with active, fun-loving, energetic 2nd graders. They are eager to learn, ask questions, and explore. One of my biggest jobs as their teacher is to keep them actively engaged in the learning experience. My students are 2nd graders who have the desire to gain as much knowledge as they can. They have a wide range of learning styles and abilities, and all have the desire to learn and create with technology. Our school is in a very diverse, middle class, hard working neighborhood. Our families value education and expect a high level of rigor. \r\nThe students in my 2nd grade classroom need a MacBook Air laptop to support their daily learning. The laptop will provide students with opportunities to take ownership and control of their own learning and also explore through technology. Having a laptop computer in the classroom will also provide them opportunities to engage in some of the lessons at their own pace and effectively collaborate with other classmates. It will give them access to up to date information quickly and easily, read and store hundreds of iBooks, allow them to work on computer coding, and use applications and software to publish completed books. The list of possibilities are endless!Mrs.Mrs.

==

My classroom consists of twenty-two amazing sixth graders from different cultures and backgrounds. They are a social bunch who enjoy working in partners and working with groups. They are hard-working and eager to head to middle school next year. My job is to get them ready to make this transition and make it as smooth as possible. In order to do this, my students need to come to school every day and feel safe and ready to learn. Because they are getting ready to head to middle school, I give them lots of choice- choice on where to sit and work, the order to complete assignments, choice of projects, etc. Part of the students feeling safe is the ability for them to come into a welcoming, encouraging environment. My room is colorful and the atmosphere is casual. I want them to take ownership of the classroom because we ALL share it together. Because my time with them is limited, I want to ensure they get the most of this time and enjoy it to the best of their abilities.Currently, we have twenty-two desks of differing sizes, yet the desks are similar to the ones the students will use in middle school. We also have a kidney table with crates for seating. I allow my students to choose their own spots while they are working independently or in groups. More often than not, most of them move out of their desks and onto the crates. Believe it or not, this has proven to be more successful than making them stay at their desks! It is because of this that I am looking toward the “Flexible Seating” option for my classroom.\r\n The students look forward to their work time so they can move around the room. I would like to get rid of the constricting desks and move toward more “fun” seating options. I am requesting variou

```
s seating so my students have more options to sit. Currently, I have a s
tool and a papasan chair I inherited from the previous sixth-grade teach
er as well as five milk crate seats I made, but I would like to give the
m more options and reduce the competition for the “good seats”. I am als
o requesting two rugs as not only more seating options but to make the c
lassroom more welcoming and appealing. In order for my students to be ab
le to write and complete work without desks, I am requesting a class set
of clipboards. Finally, due to curriculum that requires groups to work t
ogether, I am requesting tables that we can fold up when we are not usin
g them to leave more room for our flexible seating options.\r\nI know th
at with more seating options, they will be that much more excited about
coming to school! Thank you for your support in making my classroom one
students will remember forever!Mrs.Mrs.
========================================================================
=====================================================
```

In [15]:

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

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

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

In [16]:

```python
sent = decontracted(project_data['essay'].values[2020])
print(sent)
print("="*120)
```

```
I have long dreamed of teaching Angels in America, a play for my AP studen
ts that stimulated their thoughts and understand the universal promise of
the American dream.My students come from extremely diverse backgrounds.\r
\n\r\nThere are students who have fled war-torn countries such as the Ukra
ine, to first generation Mexican-Americans, to students dealing with Asber
gers and even a student who is in the advanced stages of Muscular Dystroph
y. Through all their struggles, they are extremely resilient and come to s
chool every day with hopes of a better future.In class we will be reading
\"Angels in America\" together and discussing in Socratic seminars and wri
ting papers on themes such as: visions of America, magical realism, the ne
ed for a sense of community in our lives, and how caustic and demeaning st
ereotyping can be.AP students are at an age in their lives where they are
ready to see the world through many lenses.  These unique perspectives mak
e they not only better readers and writers, but also more prepared for the
world they are entering.
==========================================================================
==============================================
```

In [17]:

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

```
I have long dreamed of teaching Angels in America a play for my AP student
s that stimulated their thoughts and understand the universal promise of t
he American dream My students come from extremely diverse backgrounds r n
r nThere are students who have fled war torn countries such as the Ukraine
to first generation Mexican Americans to students dealing with Asbergers a
nd even a student who is in the advanced stages of Muscular Dystrophy Thro
ugh all their struggles they are extremely resilient and come to school ev
ery day with hopes of a better future In class we will be reading Angels i
n America together and discussing in Socratic seminars and writing papers
on themes such as visions of America magical realism the need for a sense
of community in our lives and how caustic and demeaning stereotyping can b
e AP students are at an age in their lives where they are ready to see the
world through many lenses These unique perspectives make they not only bet
ter readers and writers but also more prepared for the world they are ente
ring
```

In [18]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ["a","about","above","after","again","against","ain","all","am","an","and","
             "as","at","be","because","been","before","being","below","between","both",
             "d","did","didn","didn't","do","does","doesn","doesn't","doing","don","don
             "for","from","further","had","hadn","hadn't","has","hasn","hasn't","have",
             "here","hers","herself","him","himself","his","how","i","if","in","into","
             "itself","just","ll","m","ma","me","mightn","mightn't","more","most","must
             "needn't","no","nor","not","now","o","of","off","on","once","only","or","o
             "out","over","own","re","s","same","shan","shan't","she","she's","should",
             "so","some","such","t","than","that","that'll","the","their","theirs","the
             "these","they","this","those","through","to","too","under","until","up","v
             "we","were","weren","weren't","what","when","where","which","while","who",
             "won't","wouldn","wouldn't","y","you","you'd","you'll","you're","you've","
             "yourselves","could","he'd","he'll","he's","here's","how's","i'd","i'll","
             "she'd","she'll","that's","there's","they'd","they'll","they're","they've"
             "what's","when's","where's","who's","why's","would","able","abst","accorda
             "across","act","actually","added","adj","affected","affecting","affects","
             "along","already","also","although","always","among","amongst","announce",
             "anymore","anyone","anything","anyway","anyways","anywhere","apparently","
             "around","aside","ask","asking","auth","available","away","awfully","b","b
             "becoming","beforehand","begin","beginning","beginnings","begins","behind"
             "beyond","biol","brief","briefly","c","ca","came","cannot","can't","cause"
             "co","com","come","comes","contain","containing","contains","couldnt","dat
             "due","e","ed","edu","effect","eg","eight","eighty","either","else","elsew
             "especially","et","etc","even","ever","every","everybody","everyone","ever
             "f","far","ff","fifth","first","five","fix","followed","following","follow
             "found","four","furthermore","g","gave","get","gets","getting","give","giv
             "gone","got","gotten","h","happens","hardly","hed","hence","hereafter","he
             "hes","hi","hid","hither","home","howbeit","however","hundred","id","ie","
             "importance","important","inc","indeed","index","information","instead","i
             "it'll","j","k","keep","keeps","kept","kg","km","know","known","knows","l"
             "later","latter","latterly","least","less","lest","let","lets","like","lik
             "'ll","look","looking","looks","ltd","made","mainly","make","makes","many"
             "meantime","meanwhile","merely","mg","might","million","miss","ml","moreov
             "mug","must","n","na","name","namely","nay","nd","near","nearly","necessar
             "neither","never","nevertheless","new","next","nine","ninety","nobody","no
             "normally","nos","noted","nothing","nowhere","obtain","obtained","obviousl
             "omitted","one","ones","onto","ord","others","otherwise","outside","overal
             "particular","particularly","past","per","perhaps","placed","please","plus
             "potentially","pp","predominantly","present","previously","primarily","pro
             "provides","put","q","que","quickly","quite","qv","r","ran","rather","rd",
             "recently","ref","refs","regarding","regardless","regards","related","rela
             "resulted","resulting","results","right","run","said","saw","say","saying"
             "seeing","seem","seemed","seeming","seems","seen","self","selves","sent","
             "shes","show","showed","shown","showns","shows","significant","significant
             "six","slightly","somebody","somehow","someone","somethan","something","so
             "somewhere","soon","sorry","specifically","specified","specify","specifyin
             "sub","substantially","successfully","sufficiently","suggest","sup","sure"
             "tends","th","thank","thanks","thanx","thats","that've","thence","thereaft
             "therein","there'll","thereof","therere","theres","thereto","thereupon","t
             "thou","though","thoughh","thousand","throug","throughout","thru","thus","
             "toward","towards","tried","tries","truly","try","trying","ts","twice","tw
             "unless","unlike","unlikely","unto","upon","ups","us","use","used","useful
             "using","usually","v","value","various","'ve","via","viz","vol","vols","vs
             "wed","welcome","went","werent","whatever","what'll","whats","whence","whe
             "whereby","wherein","wheres","whereupon","wherever","whether","whim","whit
             "who'll","whomever","whos","whose","widely","willing","wish","within","wit
             "wouldnt","www","x","yes","yet","youd","youre","z","zero","a's","ain't","a
             "appreciate","appropriate","associated","best","better","c'mon","c's","can
             "consequently","consider","considering","corresponding","course","currentl
             "entirely","exactly","example","going","greetings","hello","help","hopeful
             "indicated","indicates","inner","insofar","it'd","keep","keeps","novel","p
             "secondly","sensible","serious","seriously","sure","t's","third","thorough
             "wonder"]
```

In [19]:

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

```
Wall time: 0 ns

100%|██████████████████████████████████| 109248/109248 [06:10<00:00, 295.25
it/s]
```

In [20]:

```
# after preprocesing
preprocessed_essays[40020]
```

Out[20]:

```
'typical day classroom filled active fun loving energetic 2nd graders eage
r learn questions explore biggest jobs teacher actively engaged learning e
xperience students 2nd graders desire gain knowledge wide range learning s
tyles abilities desire learn create technology school diverse middle class
hard working neighborhood families education expect high level rigor stude
nts 2nd grade classroom macbook air laptop support daily learning laptop p
rovide students opportunities ownership control learning explore technolog
y laptop computer classroom provide opportunities engage lessons pace effe
ctively collaborate classmates access easily read store hundreds ibooks wo
rk computer coding applications software publish completed books list poss
ibilities endless'
```

1.2.2 Text Preprocessing of `project_title`

In [21]:

```
print(project_data['project_title'].tail(1))
```

```
78306    News for Kids
Name: project_title, dtype: object
```

In [22]:

```
# printing some random title texts
print(project_data['project_title'].values[19])
print('--'*19)
print(project_data['project_title'].values[196])
print('--'*19)
print(project_data['project_title'].values[1969])
print('--'*19)
print(project_data['project_title'].values[99999])
print('--'*19)
```

```
Choice Novels for Freshman Students are Needed!!!!!!
--------------------------------------
Pre-K STEM
--------------------------------------
Divide and Color!
--------------------------------------
Turning to Flexible Seating: One Sixth-Grade Class's Journey to Freedom
--------------------------------------
```

In [23]:

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

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

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

In [24]:

```
sent = decontracted(project_data['project_title'].values[99999])
print(sent)
print("="*120)
```

```
Turning to Flexible Seating: One Sixth-Grade Class is Journey to Freedom
==========================================================================
==============================================
```

In [25]:

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

```
Turning to Flexible Seating: One Sixth-Grade Class is Journey to Freedom
```

In [26]:

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

```
Turning to Flexible Seating One Sixth Grade Class is Journey to Freedom
```

In [27]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ["a","about","above","after","again","against","ain","all","am","an","and","
            "as","at","be","because","been","before","being","below","between","both",
            "d","did","didn","didn't","do","does","doesn","doesn't","doing","don","don
            "for","from","further","had","hadn","hadn't","has","hasn","hasn't","have",
            "here","hers","herself","him","himself","his","how","i","if","in","into","
            "itself","just","ll","m","ma","me","mightn","mightn't","more","most","must
            "needn't","no","nor","not","now","o","of","off","on","once","only","or","o
            "out","over","own","re","s","same","shan","shan't","she","she's","should",
            "so","some","such","t","than","that","that'll","the","their","theirs","the
            "these","they","this","those","through","to","too","under","until","up","v
            "we","were","weren","weren't","what","when","where","which","while","who",
            "won't","wouldn","wouldn't","y","you","you'd","you'll","you're","you've","
            "yourselves","could","he'd","he'll","he's","here's","how's","i'd","i'll","
            "she'd","she'll","that's","there's","they'd","they'll","they're","they've"
            "what's","when's","where's","who's","why's","would","able","abst","accorda
            "across","act","actually","added","adj","affected","affecting","affects","
            "along","already","also","although","always","among","amongst","announce",
            "anymore","anyone","anything","anyway","anyways","anywhere","apparently","
            "around","aside","ask","asking","auth","available","away","awfully","b","b
            "becoming","beforehand","begin","beginning","beginnings","begins","behind"
            "beyond","biol","brief","briefly","c","ca","came","cannot","can't","cause"
            "co","com","come","comes","contain","containing","contains","couldnt","dat
            "due","e","ed","edu","effect","eg","eight","eighty","either","else","elsew
            "especially","et","etc","even","ever","every","everybody","everyone","ever
            "f","far","ff","fifth","first","five","fix","followed","following","follow
            "found","four","furthermore","g","gave","get","gets","getting","give","giv
            "gone","got","gotten","h","happens","hardly","hed","hence","hereafter","he
            "hes","hi","hid","hither","home","howbeit","however","hundred","id","ie","
            "importance","important","inc","indeed","index","information","instead","i
            "it'll","j","k","keep","keeps","kept","kg","km","know","known","knows","l"
            "later","latter","latterly","least","less","lest","let","lets","like","lik
            "'ll","look","looking","looks","ltd","made","mainly","make","makes","many"
            "meantime","meanwhile","merely","mg","might","million","miss","ml","moreov
            "mug","must","n","na","name","namely","nay","nd","near","nearly","necessar
            "neither","never","nevertheless","new","next","nine","ninety","nobody","no
            "normally","nos","noted","nothing","nowhere","obtain","obtained","obviousl
            "omitted","one","ones","onto","ord","others","otherwise","outside","overal
            "particular","particularly","past","per","perhaps","placed","please","plus
            "potentially","pp","predominantly","present","previously","primarily","pro
            "provides","put","q","que","quickly","quite","qv","r","ran","rather","rd",
            "recently","ref","refs","regarding","regardless","regards","related","rela
            "resulted","resulting","results","right","run","said","saw","say","saying"
            "seeing","seem","seemed","seeming","seems","seen","self","selves","sent","
            "shes","show","showed","shown","showns","shows","significant","significant
            "six","slightly","somebody","somehow","someone","somethan","something","so
            "somewhere","soon","sorry","specifically","specified","specify","specifyin
            "sub","substantially","successfully","sufficiently","suggest","sup","sure"
            "tends","th","thank","thanks","thanx","thats","that've","thence","thereaft
            "therein","there'll","thereof","therere","theres","thereto","thereupon","t
            "thou","though","thoughh","thousand","throug","throughout","thru","thus","
            "toward","towards","tried","tries","truly","try","trying","ts","twice","tw
            "unless","unlike","unlikely","unto","upon","ups","us","use","used","useful
            "using","usually","v","value","various","'ve","via","viz","vol","vols","vs
            "wed","welcome","went","werent","whatever","what'll","whats","whence","whe
            "whereby","wherein","wheres","whereupon","wherever","whether","whim","whit
            "who'll","whomever","whos","whose","widely","willing","wish","within","wit
            "wouldnt","www","x","yes","yet","youd","youre","z","zero","a's","ain't","a
```

```
             "appreciate","appropriate","associated","best","better","c'mon","c's","can
             "consequently","consider","considering","corresponding","course","currentl
             "entirely","exactly","example","going","greetings","hello","help","hopeful
             "indicated","indicates","inner","insofar","it'd","keep","keeps","novel","p
             "secondly","sensible","serious","seriously","sure","t's","third","thorough
             "wonder"]
```

In [28]:

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

```
Wall time: 0 ns

100%|█████████████████████████████████| 109248/109248 [00:12<00:00, 9033.89
it/s]
```

In [29]:

```
preprocessed_project_title[99999]
```

Out[29]:

```
'turning flexible seating sixth grade class journey freedom'
```

1.3. Numerical normalization

1.3.1 normalization_price

In [30]:

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

Out[30]:

```
(109248, 20)
```

In [31]:

```
project_data.head(1)
```

Out[31]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Dat
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs	CA	2016 04-2 00:27:3

In [32]:

```
print(project_data["price"].shape)
```

```
(109248,)
```

In [33]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(1,-1)
price_normalize = Normalizer()
price_normalizer = price_normalize.fit_transform(project_data['price'].values.reshape(1
price_normalizer = price_normalizer.T
print(price_normalizer)
print("-------------------------------------------------------")
print("shape of price_normalizer:", price_normalizer.shape)
```

```
[[4.63560392e-03]
 [1.36200635e-03]
 [2.10346002e-03]
 ...
 [2.55100471e-03]
 [1.83960046e-03]
 [3.51642253e-05]]
-------------------------------------------------------
shape of price_normalizer: (109248, 1)
```

1.3.2 Normalization of teacher_number_of_previously_posted_projects

In [34]:

```
project_data['teacher_number_of_previously_posted_projects'].values
```

Out[34]:

```
array([53,  4, 10, ...,  0,  1,  2], dtype=int64)
```

In [35]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
teacher_number_of_previously_posted_projects_normalize = Normalizer()
teacher_number_of_previously_posted_projects_normalizer = teacher_number_of_previously_
teacher_number_of_previously_posted_projects_normalizer = teacher_number_of_previously_
print(teacher_number_of_previously_posted_projects_normalizer)
print("="*25)
print("Shape of teacher_number_of_previously_posted_projects_normalizer :", teacher_num
```

```
[[0.00535705]
 [0.00040431]
 [0.00101076]
 ...
 [0.        ]
 [0.00010108]
 [0.00020215]]
=========================
Shape of teacher_number_of_previously_posted_projects_normalizer : (10924
8, 1)
```

1.3.3 spilt the data into train ,CV and test

In [36]:

```
project_data.head(1)
```

Out[36]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Dat
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs	CA	2016 04-2 00:27:3

In [37]:

```
project_data['project_is_approved'].values
```

Out[37]:

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

In [38]:

```
# spliting the data into train , test CV
# Refrence link :https://scikit-learn.org/stable/modules/generated/sklearn.model_select
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data, project_data['project_
                                                    stratify= project_data['project_is_a
print("Shape of X_train and y_train  :", X_train.shape, y_train.shape)
print("Shape of X_test and y_test    :", X_test.shape, y_test.shape)
```

```
Shape of X_train and y_train  : (73196, 20) (73196,)
Shape of X_test and y_test    : (36052, 20) (36052,)
```

In [39]:

```
X_train.head(1)
```

Out[39]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
100839	179078	p245370	71c7a78d524b0c4cc829f22f7df7a47a	Ms	WI	1

In [40]:

```
X_test.head(2)
```

Out[40]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
16831	36543	p080138	49df8104ef13ae354696b275e32602e7	Mrs	MA	2
35018	72038	p108799	f19ca90a98fb6eeb66368bc7c2098f7c	Mrs	SC	2

1.4. Response coding for Categorical data

In [41]:

```
def trainResponseEncoding(running_data, cat_type):
    # alpha, the hyperparameter of Laplace smoothing, has been defaulted to 1.

    running_data.loc[running_data[cat_type].isnull(), cat_type] = 'nan' #Imputation

    data_0 = running_data[running_data['project_is_approved']==0].groupby(cat_type).siz
    data_1 = running_data[running_data['project_is_approved']==1].groupby(cat_type).siz

    return data_0, data_1

def getResponseEnconding(data_0, data_1, running_data, cat_type, alpha=1): # default al
    running_data.loc[running_data[cat_type].isnull(), cat_type] = 'nan' #if unseen data

    col_wise_dict = {'col1':[], 'col2': []}

    for row in running_data[cat_type]:
        col_wise_dict['col1'].append(data_0.get(row, 0.5))
        col_wise_dict['col2'].append(data_1.get(row, 0.5))

    class_0 = ((data_0 + alpha)/(data_0 + data_1 + alpha)).values
    class_1 = ((data_1 + alpha)/(data_0 + data_1 + alpha)).values

    response_enc = pd.DataFrame(col_wise_dict)
    print("Shape response encoding ",response_enc.shape)

    return response_enc
```

In [42]:

```
data_dict = {'X_train':{}, 'X_test': {} }
cols_dict = {    'cat_cols': ['school_state','clean_categories', 'clean_subcategories',
                          'project_grade_category','teacher_prefix']
            }

#'school_state','clean_categories', 'clean_subcategories', 'project_grade_category', 't
for col_type, cols_name in cols_dict.items():
    if col_type == 'cat_cols':
        for cat_type in cols_name:
            print (cat_type)
            data_0, data_1 = trainResponseEncoding(X_train, cat_type)
            for data_type, data_part in [('X_train', X_train), ('X_test', X_test)]:
                response_encode = getResponseEnconding(data_0, data_1, data_part, cat_t
                data_dict[data_type][cat_type] = response_encode
```

```
school_state
Shape response encoding  (73196, 2)
Shape response encoding  (36052, 2)
clean_categories
Shape response encoding  (73196, 2)
Shape response encoding  (36052, 2)
clean_subcategories
Shape response encoding  (73196, 2)
Shape response encoding  (36052, 2)
project_grade_category
Shape response encoding  (73196, 2)
Shape response encoding  (36052, 2)
teacher_prefix
Shape response encoding  (73196, 2)
Shape response encoding  (36052, 2)
```

In [190]:

```
# Accessing train cat_type
cat_0_tr              = data_dict['X_train']['clean_categories']['col1']
cat_1_tr              = data_dict['X_train']['clean_categories']['col2']
subcat_0_tr           = data_dict['X_train']['clean_subcategories']['col1']
subcat_1_tr           = data_dict['X_train']['clean_subcategories']['col2']
state_0_tr            = data_dict['X_train']['school_state']['col1']
state_1_tr            = data_dict['X_train']['school_state']['col2']
teacher_prefix_0_tr = data_dict['X_train']['teacher_prefix']['col1']
teacher_prefix_1_tr = data_dict['X_train']['teacher_prefix']['col2']
proj_grade_0_tr       = data_dict['X_train']['project_grade_category']['col1']
proj_grade_1_tr       = data_dict['X_train']['project_grade_category']['col2']
# Assessing test cat_type
cat_0_te              = data_dict['X_test']['clean_categories']['col1']
cat_1_te              = data_dict['X_test']['clean_categories']['col2']
subcat_0_te           = data_dict['X_test']['clean_subcategories']['col1']
subcat_1_te           = data_dict['X_test']['clean_subcategories']['col2']
state_0_te            = data_dict['X_test']['school_state']['col1']
state_1_te            = data_dict['X_test']['school_state']['col2']
teacher_prefix_0_te = data_dict['X_test']['teacher_prefix']['col1']
teacher_prefix_1_te = data_dict['X_test']['teacher_prefix']['col2']
proj_grade_0_te       = data_dict['X_test']['project_grade_category']['col1']
proj_grade_1_te       = data_dict['X_test']['project_grade_category']['col2']
```

In [198]:

```

normalizer = Normalizer()
cat_0_train            = normalizer.fit_transform(cat_0_tr.values.reshape(1,-1)).T
cat_1_train            = normalizer.fit_transform(cat_1_tr.values.reshape(1,-1)).T
subcat_0_train         = normalizer.fit_transform(subcat_0_tr.values.reshape(1,-1)).T
subcat_1_train         = normalizer.fit_transform(subcat_1_tr.values.reshape(1,-1)).T
state_0_train          = normalizer.fit_transform(state_0_tr.values.reshape(1,-1)).T
state_1_train          = normalizer.fit_transform(state_1_tr.values.reshape(1,-1)).T
teacher_prefix_0_train = normalizer.fit_transform(teacher_prefix_0_tr.values.reshape(1,
teacher_prefix_1_train = normalizer.fit_transform(teacher_prefix_1_tr.values.reshape(1,
proj_grade_0_train     = normalizer.fit_transform(proj_grade_0_tr.values.reshape(1,-1))
proj_grade_1_train     = normalizer.fit_transform(proj_grade_1_tr.values.reshape(1,-1))
```

In [199]:

```
normalizer = Normalizer()
cat_0_test            = normalizer.fit_transform(cat_0_te.values.reshape(1,-1)).T
cat_1_test            = normalizer.fit_transform(cat_1_te.values.reshape(1,-1)).T
subcat_0_test         = normalizer.fit_transform(subcat_0_te.values.reshape(1,-1)).T
subcat_1_test         = normalizer.fit_transform(subcat_1_te.values.reshape(1,-1)).T
state_0_test          = normalizer.fit_transform(state_0_te.values.reshape(1,-1)).T
state_1_test         = normalizer.fit_transform(state_1_te.values.reshape(1,-1)).T
teacher_prefix_0_test = normalizer.fit_transform(teacher_prefix_0_te.values.reshape(1,-
teacher_prefix_1_test = normalizer.fit_transform(teacher_prefix_1_te.values.reshape(1,-
proj_grade_0_test     = normalizer.fit_transform(proj_grade_0_te.values.reshape(1,-1)).
proj_grade_1_test     = normalizer.fit_transform(proj_grade_1_te.values.reshape(1,-1)).
```

1.5. Vectorizing Text

1.5.1 Vectorization of essays bow

In [200]:

```
X_train['essay'].tail(1)
```

Out[200]:

```
99801    I teach social skills to a population of stude...
Name: essay, dtype: object
```

In [201]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_essays_bow = CountVectorizer(preprocessed_essays, min_df=10,  ngram_range=(1

essays_bow_train = vectorizer_essays_bow.fit_transform(X_train['essay'].values)
essays_bow_test  = vectorizer_essays_bow.transform(X_test['essay'].values)

print("---------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",essays_bow_train.shape)
print("Shape of matrix after one hot encodig test  : ",essays_bow_test.shape)
```

```
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (73196, 14782)
Shape of matrix after one hot encodig test  :  (36052, 14782)
```

In [202]:

```
print(vectorizer_essays_bow.get_feature_names)
```

```
<bound method CountVectorizer.get_feature_names of CountVectorizer(analyze
r='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8',
                input=['fortunate fairy tale stem kits classroom stem jour
nals '
                       'students enjoyed love implement lakeshore stem kit
s '
                       'classroom school year provide excellent engaging s
tem '
                       'lessons students variety backgrounds including '
                       'language socioeconomic status lot experience sc...
                       'requesting games practice phonics skills imagine r
ead '
                       'text presented clue read frustration intensified h
igh '
                       'stakes testing live support struggling readers lon
ger '
                       'struggling successful learners readers', ...],
                lowercase=True, max_df=1.0, max_features=None, min_df=10,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)>
```

1.5.1.1Vectorization of essay tfidf

In [203]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_essays_tfidf = TfidfVectorizer(preprocessed_essays, min_df=10, ngram_range=(

essays_tfidf_train = vectorizer_essays_tfidf.fit_transform(X_train['essay'].values)
essays_tfidf_test  = vectorizer_essays_tfidf.transform(X_test['essay'].values)

print("Shape of matrix after one hot encodig of train : ",essays_tfidf_train.shape)
print("Shape of matrix after one hot encodig test     : ",essays_tfidf_test.shape)
```

```
Shape of matrix after one hot encodig of train :  (73196, 14782)
Shape of matrix after one hot encodig test     :  (36052, 14782)
```

1.5.1.2 Using Pretrained Models: essays Avg W2V

In [204]:

```
'''
# Reading glove vectors in python: https://stackoverflow.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 val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

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

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

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

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

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

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


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

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


'''
```

Out[204]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/382303
```

```
49/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) load
GloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open
(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm
(f):\n        splitLine = line.split()\n        word = splitLine[0]\n
embedding = np.array([float(val) for val in splitLine[1:]])\n        mod
el[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n
return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ======
======================\nOutput:\n    \nLoading Glove Model\n1917495it [0
6:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ==================
==========\n\nwords = []\nfor i in preproced_essays:\n    words.extend
(i.split(\' \'))\n\nfor i in preprocessed_project_title:\n    words.exte
nd(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwo
rds = set(words)\nprint("the unique words in the coupus", len(words))\n
\ninter_words = set(model.keys()).intersection(words)\nprint("The number
of words that are present in both glove vectors and our coupus",       l
en(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 python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-
load-variables-in-python/\n\nimport (http://www.jessicayung.com/how-to-u
se-pickle-to-save-and-load-variables-in-python/\n\nimport) pickle\nwith
 open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus,
 f)\n\n\n'
```

In [205]:

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

In [206]:

```python
np
import numpy as np
```

In [207]:

```
# average Word2Vec X_train
# compute average word2vec for each review.
essays_avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in
for sentence in tqdm(X_train['essay']): # 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
    essays_avg_w2v_vectors_train.append(vector)

print(len(essays_avg_w2v_vectors_train))
print(len(essays_avg_w2v_vectors_train[0]))
```

```
  0%|                                                | 0/73196 [00:00<?, 
?it/s]

  0%|                                      | 93/73196 [00:00<01:18, 929.
96it/s]

  0%|                                     | 186/73196 [00:00<01:18, 929.
95it/s]

  0%|▌                                    | 298/73196 [00:00<01:14, 97
9.82it/s]

  1%|▌                                    | 392/73196 [00:00<01:15, 96
4.53it/s]

  1%|▌                                    | 488/73196 [00:00<01:15, 96
3.15it/s]
```

average Word2Vec X_test_essay

In [208]:

```
# average Word2Vec X_test_essay
# compute average word2vec for each review.
essays_avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in th
for sentence in tqdm(X_test['essay']): # 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
    essays_avg_w2v_vectors_test.append(vector)

print(len(essays_avg_w2v_vectors_test))
print(len(essays_avg_w2v_vectors_test[0]))
```

```
  0%|                                                | 0/36052 [00:00<?,
?it/s]

  0%|                                      | 84/36052 [00:00<00:42, 839.
95it/s]

  1%|▏                                    | 198/36052 [00:00<00:39, 91
1.95it/s]

  1%|▍                                    | 299/36052 [00:00<00:38, 93
9.29it/s]

  1%|▌                                    | 397/36052 [00:00<00:37, 95
1.13it/s]

  1%|▌                                   | 490/36052 [00:00<00:37, 944.
67it/s]
```

In [209]:

```
type(essays_avg_w2v_vectors_train)
```

Out[209]:

```
list
```

1.5.1.3 essays TFIDF weighted W2V train

In [210]:

```
tfidf_model_preprocessed_essays_train = TfidfVectorizer()
tfidf_model_preprocessed_essays_train.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_essays_train.get_feature_names(), list(t
tfidf_words = set(tfidf_model_preprocessed_essays_train.get_feature_names())
```

In [211]:

```
# essays TFIDF weighted W2V_train
# compute average word2vec for each review.
preprocessed_essays_train_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/revie
for sentence in tqdm(X_train['essay']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_essays_train_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_essays_train_tfidf_w2v_vectors))
print(len(preprocessed_essays_train_tfidf_w2v_vectors[0]))
```

```
  0%|                                                | 0/73196 [00:00<?,
?it/s]

  0%|                                        | 7/73196 [00:00<18:07, 67.
30it/s]

  0%|                                       | 16/73196 [00:00<17:02, 71.
60it/s]

  0%|                                       | 26/73196 [00:00<15:41, 77.
72it/s]

  0%|                                       | 33/73196 [00:00<16:18, 74.
74it/s]

  0%|                                       | 41/73196 [00:00<15:59, 76.
25it/s]
```

In [212]:

```
# tfidf_model_preprocessed_essays_test
# compute average word2vec for each review.
preprocessed_essays_test_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review
for sentence in tqdm(X_test['essay']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_essays_test_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_essays_test_tfidf_w2v_vectors))
print(len(preprocessed_essays_test_tfidf_w2v_vectors[0]))
```

```
  0%|                                                  | 0/36052 [00:00<?,
?it/s]

  0%|                                        | 11/36052 [00:00<05:44, 104.
76it/s]

  0%|                                         | 22/36052 [00:00<06:02, 99.
36it/s]

  0%|                                         | 32/36052 [00:00<06:03, 98.
96it/s]

  0%|                                         | 40/36052 [00:00<06:31, 92.
07it/s]

  0%|                                         | 49/36052 [00:00<06:36, 90.
88it/s]
```

1.5.2 Vectorization of project_title bow train, test

In [60]:

```
X_train['project_title'].head(1)
```

Out[60]:

```
100839    Lights, Camera...Action!
Name: project_title, dtype: object
```

In [61]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_project_title_bow = CountVectorizer(preprocessed_project_title, min_df=10,ng

project_title_bow_train = vectorizer_project_title_bow.fit_transform(X_train['project_t
project_title_bow_test  = vectorizer_project_title_bow.transform(X_test['project_title'

print("Shape of matrix after one hot encodig train : ",project_title_bow_train.shape)
print("Shape of matrix after one hot encodig test  : ",project_title_bow_test.shape)
```

```
Shape of matrix after one hot encodig train :  (73196, 2641)
Shape of matrix after one hot encodig test  :  (36052, 2641)
```

1.5.2.1 Vectorization of project_title tfidf train, test

In [62]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_project_title_tfidf = TfidfVectorizer(preprocessed_project_title, min_df=10,

project_title_tfidf_train = vectorizer_project_title_tfidf.fit_transform(X_train['proje
project_title_tfidf_test  = vectorizer_project_title_tfidf.transform(X_test['project_ti

print("Shape of matrix after one hot encodig of train : ",project_title_tfidf_train.sha
print("Shape of matrix after one hot encodig test     : ",project_title_tfidf_test.shap
```

```
Shape of matrix after one hot encodig of train :  (73196, 2641)
Shape of matrix after one hot encodig test     :  (36052, 2641)
```

1.5.2.2 Using Pretrained Models: project_title Avg W2V train

In [63]:

```
'''
# Reading glove vectors in python: https://stackoverflow.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 val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

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

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

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

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

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

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


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

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


'''
```

Out[63]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/382303
```

```
49/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) load
GloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open
(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm
(f):\n        splitLine = line.split()\n        word = splitLine[0]\n
embedding = np.array([float(val) for val in splitLine[1:]])\n        mod
el[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n
return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ======
=====================\nOutput:\n    \nLoading Glove Model\n1917495it [0
6:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ==================
==========\n\nwords = []\nfor i in preproced_essays:\n    words.extend
(i.split(\' \'))\n\nfor i in preprocessed_project_title:\n    words.exte
nd(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwo
rds = set(words)\nprint("the unique words in the coupus", len(words))\n
\ninter_words = set(model.keys()).intersection(words)\nprint("The number
of words that are present in both glove vectors and our coupus",       l
en(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 python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-
load-variables-in-python/\n\nimport (http://www.jessicayung.com/how-to-u
se-pickle-to-save-and-load-variables-in-python/\n\nimport) pickle\nwith
 open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus,
 f)\n\n\n'
```

In [64]:

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

In [65]:

```
# average Word2Vec  project_title_train
# compute average word2vec for each review.
project_title_avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is sto
for sentence in tqdm(X_train['project_title']): # 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
    project_title_avg_w2v_vectors_train.append(vector)

print(len(project_title_avg_w2v_vectors_train))
print(len(project_title_avg_w2v_vectors_train[0]))
```

```
  0%|                                                | 0/73196 [00:00<?, ?
it/s]

  8%|██▊                                 | 5668/73196 [00:00<00:01, 56676.8
1it/s]

 14%|████▉                              | 10392/73196 [00:00<00:01, 53471.2
1it/s]

 22%|███████▋                           | 15887/73196 [00:00<00:01, 53905.5
3it/s]

 29%|██████████▏                       | 21249/73196 [00:00<00:00, 53818.64
it/s]

 36%|████████████▋                      | 26442/73196 [00:00<00:00, 53236.8
2it/s]

 43%|███████████████                    | 31622/73196 [00:00<00:00, 52796.5
6it/s]

 51%|█████████████████▊                 | 37222/73196 [00:00<00:00, 53717.5
7it/s]

 58%|████████████████████▏              | 42139/73196 [00:00<00:00, 52100.2
1it/s]

 65%|██████████████████████▋            | 47475/73196 [00:00<00:00, 52470.9
9it/s]

 72%|█████████████████████████          | 52508/73196 [00:01<00:00, 49515.0
9it/s]

 78%|███████████████████████████▍       | 57344/73196 [00:01<00:00, 45040.8
4it/s]

 84%|█████████████████████████████▌     | 61845/73196 [00:01<00:00, 39360.0
1it/s]

 90%|███████████████████████████████▌   | 65908/73196 [00:01<00:00, 38820.7
6it/s]

100%|█████████████████████████████████| 73196/73196 [00:01<00:00, 45630.81
it/s]

73196
300
```

In [66]:

```
# average Word2Vec  project_title_test
# compute average word2vec for each review.
project_title_avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stor
for sentence in tqdm(X_test['project_title']): # 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
    project_title_avg_w2v_vectors_test.append(vector)

print(len(project_title_avg_w2v_vectors_test))
print(len(project_title_avg_w2v_vectors_test[0]))
```

```
  0%|                                                | 0/36052 [00:00<?, ?
it/s]

 11%|███▋                              | 3797/36052 [00:00<00:00, 37967.77
it/s]

 23%|███████▋                           | 8299/36052 [00:00<00:00, 39839.4
2it/s]

 35%|███████████▋                      | 12745/36052 [00:00<00:00, 41120.8
6it/s]

 48%|███████████████▋                  | 17152/36052 [00:00<00:00, 41962.5
8it/s]

 58%|███████████████████▏              | 21012/36052 [00:00<00:00, 40893.1
3it/s]

 71%|███████████████████████▍          | 25451/36052 [00:00<00:00, 41882.2
4it/s]

 84%|███████████████████████████▋      | 30270/36052 [00:00<00:00, 43593.4
3it/s]

100%|█████████████████████████████████| 36052/36052 [00:00<00:00, 43381.38
it/s]

36052
300
```

1.5.2.3 project_title TFIDF weighted W2V train

In [67]:

```
tfidf_model_preprocessed_project_title_train = TfidfVectorizer()
tfidf_model_preprocessed_project_title_train.fit(X_train['project_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_project_title_train.get_feature_names(),
tfidf_words = set(tfidf_model_preprocessed_project_title_train.get_feature_names())
```

In [68]:

```
# project_title TFIDF weighted W2V train
# compute average word2vec for each review.
preprocessed_project_title_train_tfidf_w2v_vectors = []; # the avg-w2v for each sentenc
for sentence in tqdm(X_train['project_title']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_project_title_train_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_project_title_train_tfidf_w2v_vectors))
print(len(preprocessed_project_title_train_tfidf_w2v_vectors[0]))
```

```
  0%|                                                  | 0/73196 [00:00<?, ?
it/s]

  4%|█▍                                 | 2835/73196 [00:00<00:02, 28348.4
0it/s]

  8%|██▌                               | 5614/73196 [00:00<00:02, 28178.06
it/s]

 11%|████▍                              | 8340/73196 [00:00<00:02, 27895.7
1it/s]

 15%|█████▋                            | 10983/73196 [00:00<00:02, 27438.7
3it/s]

 19%|██████▎                           | 13794/73196 [00:00<00:02, 27636.2
6it/s]

 22%|███████▌                          | 16283/73196 [00:00<00:02, 26750.2
6it/s]

 26%|████████▌                        | 19024/73196 [00:00<00:02, 26944.38
it/s]

 30%|██████████▍                       | 21834/73196 [00:00<00:01, 27280.4
9it/s]

 33%|███████████                      | 24431/73196 [00:00<00:01, 26873.20
it/s]

 37%|████████████▌                     | 27211/73196 [00:01<00:01, 27144.2
4it/s]

 41%|█████████████▌                   | 30220/73196 [00:01<00:01, 27965.13
it/s]
```

```
 45%|██████████████▉                  | 33240/73196 [00:01<00:01, 28599.6
0it/s]

 50%|████████████████▍                | 36353/73196 [00:01<00:01, 29313.9
8it/s]

 54%|█████████████████▋               | 39269/73196 [00:01<00:01, 28662.9
5it/s]

 58%|██████████████████▉              | 42128/73196 [00:01<00:01, 28133.2
0it/s]

 61%|████████████████████▎            | 44999/73196 [00:01<00:00, 28303.3
2it/s]

 65%|█████████████████████▌           | 47828/73196 [00:01<00:00, 27236.22
it/s]

 69%|██████████████████████▊          | 50559/73196 [00:01<00:00, 27257.8
5it/s]

 73%|████████████████████████         | 53337/73196 [00:01<00:00, 27411.95
it/s]

 77%|█████████████████████████▎       | 56082/73196 [00:02<00:00, 26547.6
3it/s]

 80%|██████████████████████████▍      | 58746/73196 [00:02<00:00, 25802.6
1it/s]

 84%|███████████████████████████▋     | 61387/73196 [00:02<00:00, 25981.4
2it/s]

 87%|████████████████████████████▊    | 63993/73196 [00:02<00:00, 25470.7
3it/s]

 91%|██████████████████████████████   | 66548/73196 [00:02<00:00, 25118.04
it/s]

 95%|███████████████████████████████▎ | 69381/73196 [00:02<00:00, 26002.0
6it/s]

100%|█████████████████████████████████| 73196/73196 [00:02<00:00, 26968.25
it/s]

73196
300
```

In [69]:

```
# project_title TFIDF weighted W2V_ test
# compute average word2vec for each review.
preprocessed_project_title_test_tfidf_w2v_vectors = []; # the avg-w2v for each sentence,
for sentence in tqdm(X_test['project_title']): # 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((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_project_title_test_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_project_title_test_tfidf_w2v_vectors))
print(len(preprocessed_project_title_test_tfidf_w2v_vectors[0]))
```

```
  0%|                                                | 0/36052 [00:00<?,
?it/s]

  8%|███▌                                 | 3051/36052 [00:00<00:01, 3050
8.28it/s]

 15%|█████▏                               | 5482/36052 [00:00<00:01, 2833
9.92it/s]

 23%|███████▉                             | 8124/36052 [00:00<00:01, 2773
4.78it/s]

 30%|██████████▌                         | 10842/36052 [00:00<00:00, 2756
5.49it/s]

 38%|█████████████▏                      | 13818/36052 [00:00<00:00, 2818
8.64it/s]

 45%|███████████████▌                    | 16371/36052 [00:00<00:00, 2733
4.17it/s]

 52%|█████████████████▏                  | 18790/36052 [00:00<00:00, 2630
7.82it/s]

 59%|███████████████████▌               | 21401/36052 [00:00<00:00, 26247.
71it/s]

 66%|██████████████████████▍             | 23864/36052 [00:00<00:00, 2495
7.54it/s]

 74%|████████████████████████▌           | 26565/36052 [00:01<00:00, 2553
9.35it/s]

 82%|███████████████████████████        | 29514/36052 [00:01<00:00, 26608.
33it/s]
```

```
 90%|█████████████████████████████▉    | 32481/36052 [00:01<00:00, 2745
7.93it/s]

100%|█████████████████████████████████| 36052/36052 [00:01<00:00, 26743.
28it/s]

36052
300
```

1.6. Vectorizing Numerical features

1.6.1 Normalization of price_train_test_cv

In [70]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(1, -1)
price_normalizer = Normalizer()
price_normalizer_train = price_normalizer.fit_transform(X_train['price'].values.reshape
price_normalizer_test  = price_normalizer.transform(X_test['price'].values.reshape(1,-1

print("shape of price_normalizer_train:", price_normalizer_train.shape)
print("----------------------------")
print(price_normalizer_train)

print("shape of price_normalizer_test :", price_normalizer_test.shape)
print("----------------------------")
print(price_normalizer_test)
```

```
shape of price_normalizer_train: (73196, 1)
----------------------------
[[0.00309706]
 [0.00149629]
 [0.00139818]
 ...
 [0.00045795]
 [0.00254776]
 [0.00177918]]
shape of price_normalizer_test : (36052, 1)
----------------------------
[[3.93945532e-04]
 [2.74233296e-03]
 [1.42573440e-03]
 ...
 [7.01346682e-05]
 [3.03624668e-03]
 [6.92355058e-04]]
```

In [71]:

```
# reshape (-1,1 ) # -1=Any row and 1=column
price_normalizer_2 = Normalizer()
price_normalizer_train_2 = price_normalizer_2.fit_transform(X_train['price'].values.res
price_normalizer_test_2  = price_normalizer_2.transform(X_test['price'].values.reshape(
```

1.6.2 Teacher_number_of_previously_posted_projects_train_test_cv : Numerical / Normalization

In [72]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(-1, 1)
teacher_number_of_previously_posted_projects_normalizer = Normalizer()

teacher_number_of_previously_posted_projects_normalizer_train = teacher_number_of_previ
teacher_number_of_previously_posted_projects_normalizer_test  = teacher_number_of_previ
print("shape of teacher_number_of_previously_posted_projects_normalizer_train:",teacher_
print("---------------------------")
print(teacher_number_of_previously_posted_projects_normalizer_train)

print("shape of teacher_number_of_previously_posted_projects_normalizer_test :",teacher_
print("---------------------------")
print(teacher_number_of_previously_posted_projects_normalizer_test)
```

```
shape of teacher_number_of_previously_posted_projects_normalizer_train: (7
3196, 1)
---------------------------
[[0.00061729]
 [0.        ]
 [0.0086421 ]
 ...
 [0.00049383]
 [0.00012346]
 [0.        ]]
shape of teacher_number_of_previously_posted_projects_normalizer_test : (3
6052, 1)
---------------------------
[[0.00140821]
 [0.        ]
 [0.00510477]
 ...
 [0.        ]
 [0.00017603]
 [0.01302596]]
```

In [ ]:

In [75]:

```
import numpy
essays_avg_w2v_vec_train = numpy.array(essays_avg_w2v_vectors_train)
essays_avg_w2v_vec_test = numpy.array(essays_avg_w2v_vectors_test)
project_title_avg_w2v_vec_train = numpy.array(project_title_avg_w2v_vectors_train)
project_title_avg_w2v_vec_test = numpy.array(project_title_avg_w2v_vectors_test)
essays_train_tfidf_w2v_vec = numpy.array(preprocessed_essays_train_tfidf_w2v_vectors)
essays_test_tfidf_w2v_vec = numpy.array(preprocessed_essays_test_tfidf_w2v_vectors)
project_title_train_tfidf_w2v_vec = numpy.array(preprocessed_project_title_train_tfidf_
project_title_test_tfidf_w2v_vec = numpy.array(preprocessed_project_title_test_tfidf_w2
print("Shape of essays avg w2v train         :",essays_avg_w2v_vec_train.shape)
print("Shape of essays avg w2v test          :",essays_avg_w2v_vec_test.shape)
print("Shape of essays tfidf w2v train       :",essays_train_tfidf_w2v_vec.shape)
print("Shape of essays tfidf w2v test        :",essays_test_tfidf_w2v_vec.shape)
print("Shape of project title avg w2v train  :",project_title_avg_w2v_vec_train.shape)
print("Shape of project title avg w2v test   :",project_title_avg_w2v_vec_test.shape)
print("Shape of project title tfidf w2v train:",project_title_train_tfidf_w2v_vec.shape
print("Shape of project title tfidf w2v test :",project_title_test_tfidf_w2v_vec.shape)
```

```
Shape of essays avg w2v train         : (73196, 300)
Shape of essays avg w2v test          : (36052, 300)
Shape of essays tfidf w2v train       : (73196, 300)
Shape of essays tfidf w2v test        : (36052, 300)
Shape of project title avg w2v train  : (73196, 300)
Shape of project title avg w2v test   : (36052, 300)
Shape of project title tfidf w2v train: (73196, 300)
Shape of project title tfidf w2v test : (36052, 300)
```

In [214]:

```
project_data.columns
```

Out[214]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_title', 'project_essay_
1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay', 'price',
       'quantity'],
      dtype='object')
```

In [215]:

```
X_train.columns
```

Out[215]:

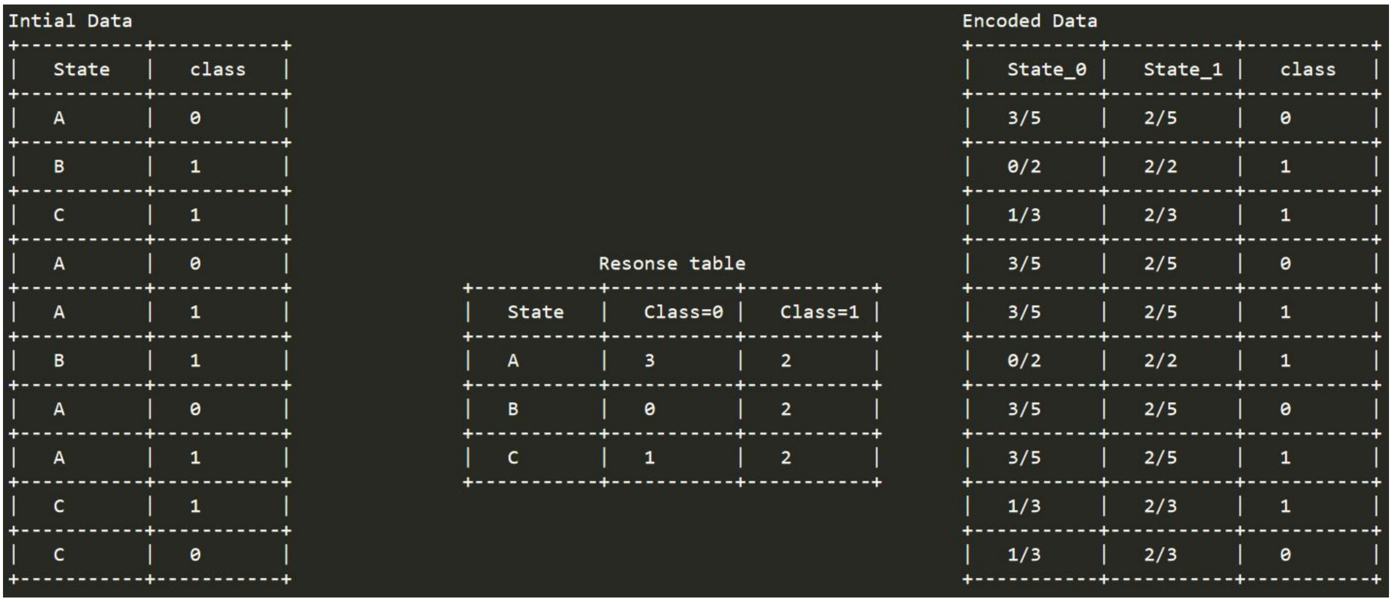
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_title', 'project_essay_
1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay', 'price',
       'quantity'],
      dtype='object')
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

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

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

Assignment 9: RF and GBDT

Response Coding: Example



Intial Data

State	class
A	0
B	1
C	1
A	0
A	1
B	1
A	0
A	1
C	1
C	0

Resonse table

State	Class=0	Class=1
A	3	2
B	0	2
C	1	2

Encoded Data

State_0	State_1	class
3/5	2/5	0
0/2	2/2	1
1/3	2/3	1
3/5	2/5	0
3/5	2/5	1
0/2	2/2	1
3/5	2/5	0
3/5	2/5	1
1/3	2/3	1
1/3	2/3	0

> 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, 0.5]

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

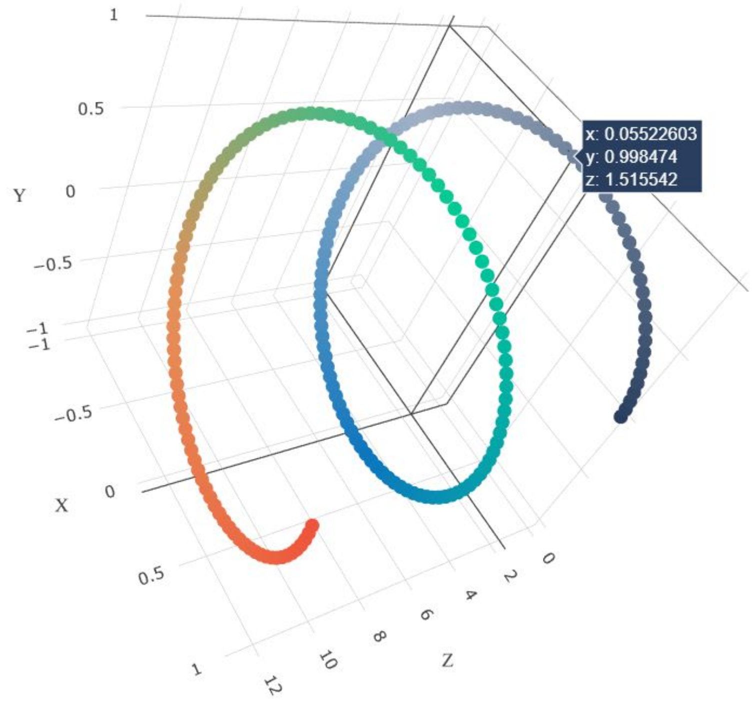
    - Set 1: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
    - Set 3: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V). Here for this set take **20K** datapoints only.
    - Set 4: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V). Here for this set take **20K** datapoints only.

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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Find the best hyper paramter using simple cross validation data
    - You can write your own for loops to do this task

3. **Representation of results**

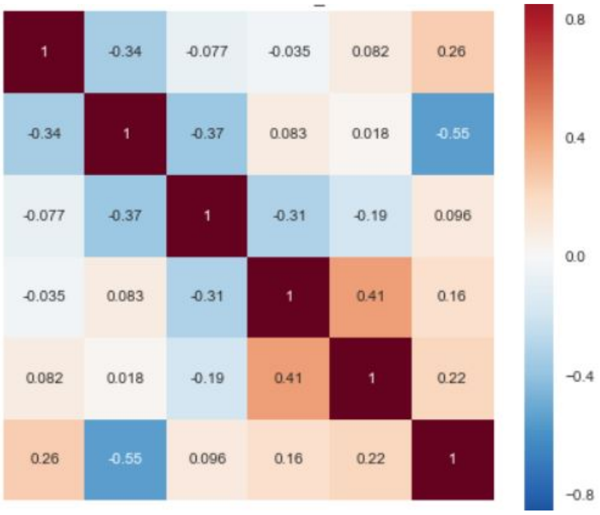
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



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

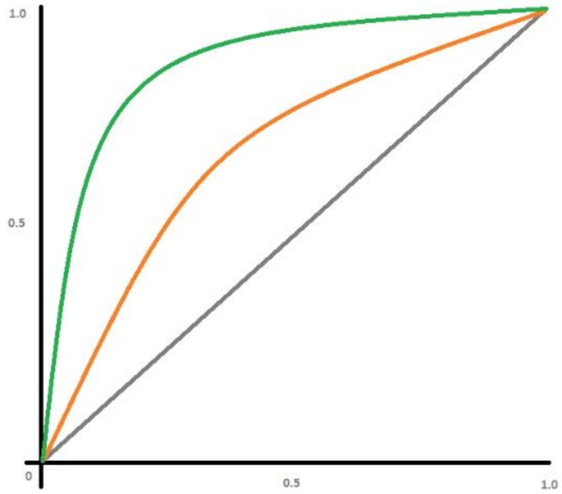
or

- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure

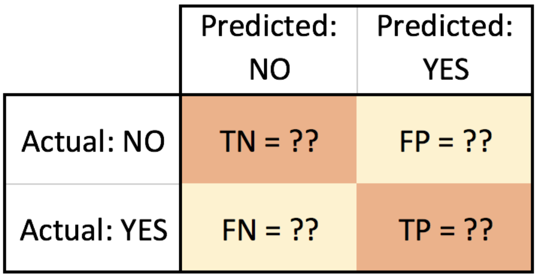


[seaborn heat maps (https://seaborn.pydata.org/generated/seaborn.heatmap.html)](https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**

- You can choose either of the plotting techniques: 3d plot or heat map
- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.



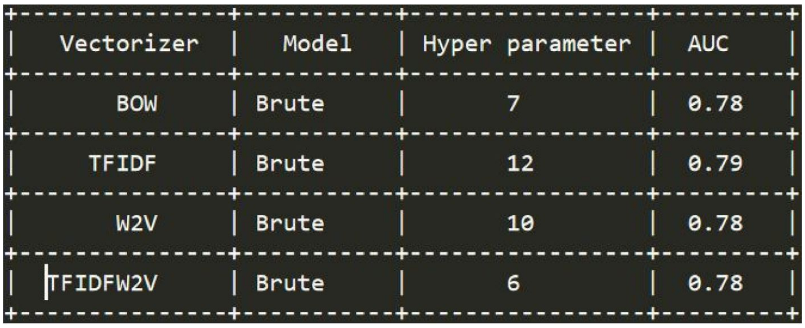
- Along with plotting ROC curve, you need to print the [confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points



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

4. **Conclusion**

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



```
+----------------+-----------+------------------+----------+
|   Vectorizer   |   Model   | Hyper parameter  |   AUC    |
+----------------+-----------+------------------+----------+
|      BOW       | Brute     |        7         |  0.78    |
+----------------+-----------+------------------+----------+
|     TFIDF      | Brute     |        12        |  0.79    |
+----------------+-----------+------------------+----------+
|      W2V       | Brute     |        10        |  0.78    |
+----------------+-----------+------------------+----------+
|   TFIDFW2V     | Brute     |        6         |  0.78    |
+----------------+-----------+------------------+----------+
```

2. Random Forest

2.1 Applying Random Forest on BOW, SET 1

Merging features encoding numerical + categorical features BOW, SET 1

In [216]:

```
#merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set1 = all categorical features + numarical features + essays_bow + project_title_bow
from scipy.sparse import hstack

set1_train = hstack((cat_0_train,cat_1_train, subcat_0_train, subcat_1_train,state_0_tr
                     teacher_prefix_1_train,teacher_prefix_0_train,proj_grade_0_train,p
                     essays_bow_train,project_title_bow_train,teacher_number_of_previou
                     price_normalizer_train)).tocsr()

set1_test = hstack((cat_0_test,cat_1_test, subcat_0_test, subcat_1_test,state_0_test,st
                     teacher_prefix_1_test,proj_grade_0_test,proj_grade_1_test,project_
                     essays_bow_test,teacher_number_of_previously_posted_projects_norma
                     price_normalizer_test)).tocsr()

print("Final Data Matrix of set1 :")
print("shape of set1_train and y_train :", set1_train.shape , y_train.shape)
print("shape of set1_test and y_test   :", set1_test.shape , y_test.shape)
```

```
Final Data Matrix of set1 :
shape of set1_train and y_train : (73196, 17435) (73196,)
shape of set1_test and y_test   : (36052, 17435) (36052,)
```

2.1.1 Hyper parameter Tuning to find best estimator :: set1_Bow

In [217]:

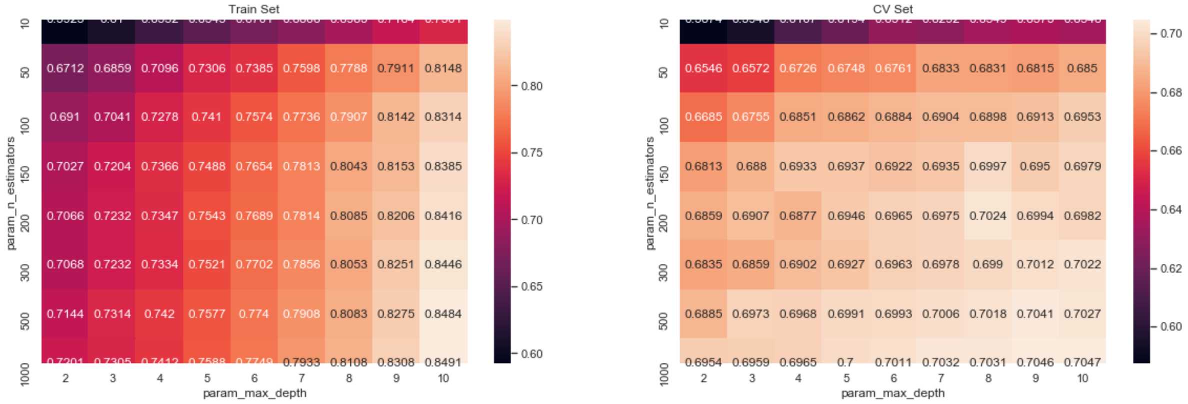
```
%%time
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
rfc1 = RandomForestClassifier(class_weight = 'balanced')
parameters = {'max_depth': [2,3,4,5,6,7,8,9,10], 'n_estimators': [10, 50, 100, 150, 200
clf1 = GridSearchCV(rfc1, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf1_fit = clf1.fit(set1_train, y_train)
```

```
Wall time: 1h 53min 26s
```

In [218]:

```
%time
# ploting the performance of model both on train data and cross validation data for eac
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(clf1.cv_results_).groupby(['param_n_estimators', 'param_max_

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()
```

```
Wall time: 0 ns
```



In [219]:

```
# Best tune parameters with score
print(clf1.best_estimator_)
print(clf1.score(set1_train,y_train))
print(clf1.score(set1_test,y_test))
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=10, max_features='aut
o',
                       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)
0.8168056900668559
0.5633014883365468
```

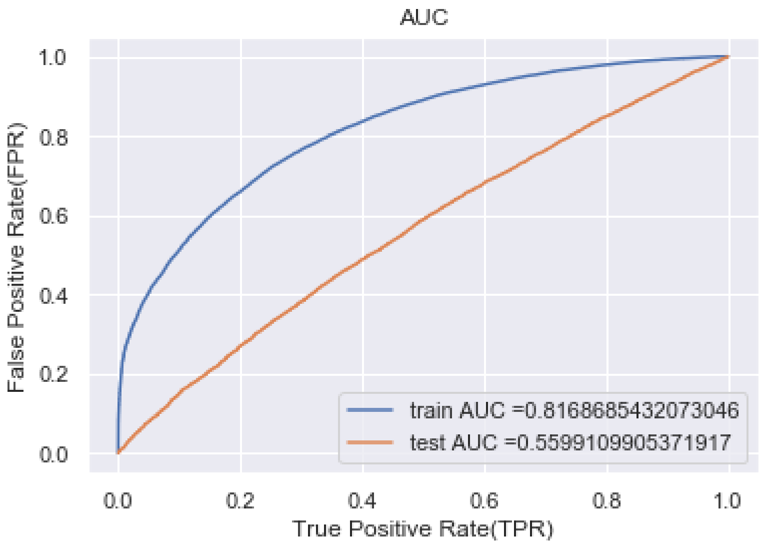
In [479]:

```
# best tune parameters
best_tune_parameters=[{'max_depth':[10], 'n_estimators':[100]}]
```

2.1.2.Train model using the best hyper-parameter value set1_bow

In [249]:

```
%%time
from sklearn.metrics import auc,roc_curve

clf_1v =RandomForestClassifier (class_weight = 'balanced',max_depth=10,n_estimators=100
clf_1v.fit(set1_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_1v.predict_proba(set1_train)[:
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_1v.predict_proba(set1_test)[:,1]

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



```
Wall time: 4min 54s
Parser   : 339 ms
```

2.1.3. Confustion Matrix set1_train and set1_test

In [250]:

```
#del np
#import numpy as np
```

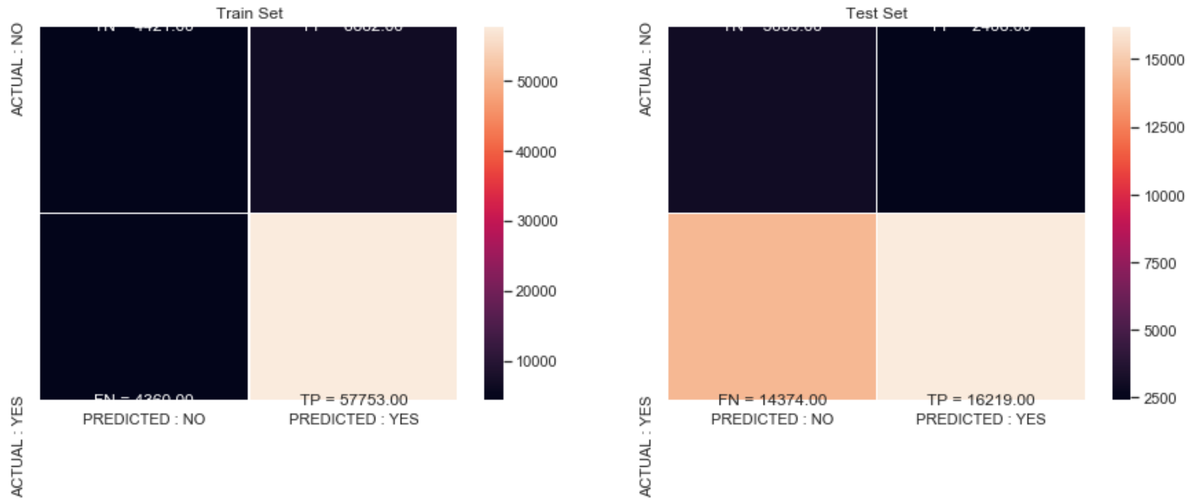
In [251]:

```
#Confusion Matrix

def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(fpr*(1-tpr))]
    print("the maximum value of tpr*(1-fpr)", np.round(max(tpr*(1-fpr)),2) , "for thres
    predictions = []
    global predictions1 # make it global
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    predictions1= predictions
    return predictions
```

In [252]:

```
clf_1v.fit(set1_train,y_train)
y_train_pred_1 = clf_1v.predict_proba(set1_train)[:,1]
y_test_pred_1 = clf_1v.predict_proba(set1_test)[:,1]


#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
conf_matr_df_train_1 = confusion_matrix(y_train, predict(y_train_pred_1, thresholds, tr
conf_matr_df_test_1 = confusion_matrix(y_test, predict(y_test_pred_1, thresholds, test_

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k

labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(ke

sns.heatmap(conf_matr_df_train_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''

sns.heatmap(conf_matr_df_test_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.54 for threshold 0.48
the maximum value of tpr*(1-fpr) 0.3 for threshold 0.48
```



In [253]:

```
# Confusion Matrix of Test
print(conf_matr_df_test_1)
```

```
[[ 3053  2406]
 [14374 16219]]
```

2.2.1 Applying Random Forest on Tfidf, SET 2

In [220]:

```
#merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set2_ = all categorical features + numarical features + essays_tfidf + project_title_

set2_train = hstack((cat_0_train,cat_1_train, subcat_0_train, subcat_1_train,state_0_tr
                     teacher_prefix_1_train,teacher_prefix_0_train,proj_grade_0_train,p
                     essays_tfidf_train,teacher_number_of_previously_posted_projects_no
                     project_title_tfidf_train,price_normalizer_train)).tocsr()


set2_test = hstack((cat_0_test,cat_1_test, subcat_0_test, subcat_1_test,state_0_test,st
                    teacher_prefix_1_test,teacher_prefix_0_test,proj_grade_0_test,proj_
                    essays_tfidf_test,teacher_number_of_previously_posted_projects_norm
                    project_title_tfidf_test,price_normalizer_test)).tocsr()

print("Final Data Matrix of set2 :")
print("shape of set2_train and y_train :", set2_train.shape , y_train.shape)
print("shape of set2_test and y_test   :", set2_test.shape , y_test.shape)
```

```
Final Data Matrix of set2 :
shape of set2_train and y_train : (73196, 17435) (73196,)
shape of set2_test and y_test   : (36052, 17435) (36052,)
```

2.2.1 Hyperparameter Tuning to find the best estimator :: set2_GridSearchCV

In [221]:

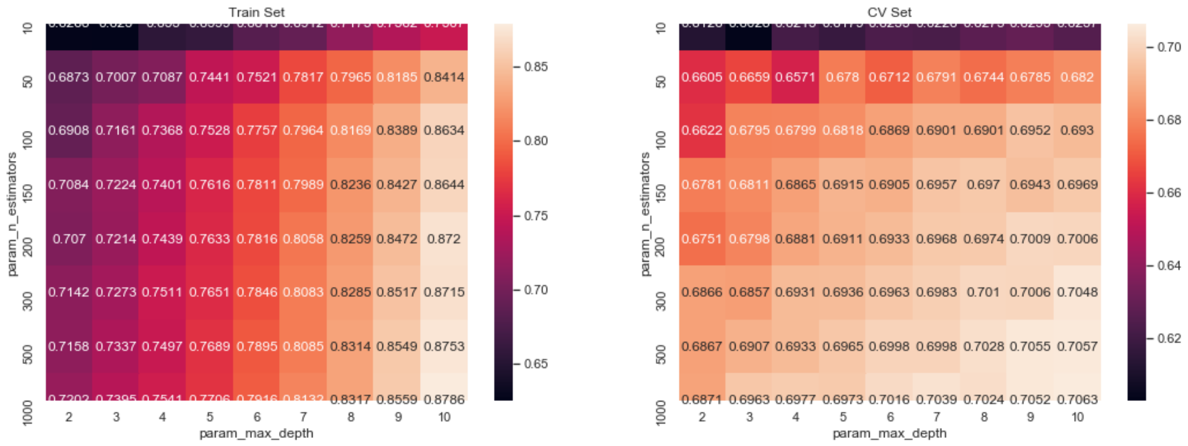
```
%time
rfc2 = RandomForestClassifier(class_weight = 'balanced')
parameters = {'max_depth': [2,3,4,5,6,7,8,9,10], 'n_estimators': [10, 50, 100, 150, 200
clf2 = GridSearchCV(rfc2, parameters, cv=3, scoring='roc_auc',return_train_score=True)
set2_fit = clf2.fit(set2_train, y_train)
```

```
Wall time: 0 ns
```

In [222]:

```
%time
import seaborn as sns; sns.set()
max_scores2 = pd.DataFrame(clf2.cv_results_).groupby(['param_n_estimators', 'param_max_

fig, ax = plt.subplots(1,2, figsize=(18,6))
sns.heatmap(max_scores2.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores2.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```

```
Wall time: 0 ns
```



In [223]:

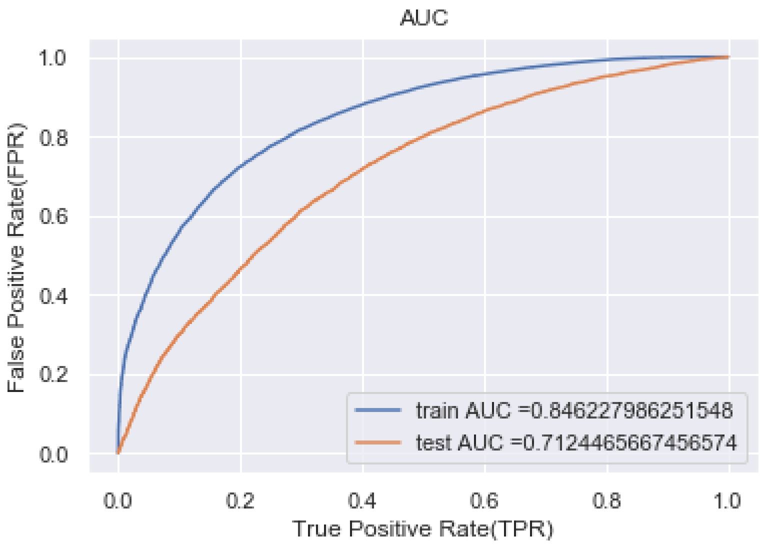
```
#Best Estimator and Best tune parameters
print(clf2.best_estimator_)
#Mean cross-validated score of the best_estimator
print(clf2.score(set2_train,y_train))
print(clf2.score(set2_test,y_test))
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=10, max_features='aut
o',
                       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)
0.847100690224025
0.7129982196514693
```

2.2.2 Train model using best estimator : set2_tfidf

In [254]:

```
%%time
from sklearn.metrics import auc,roc_curve

clf_2v =RandomForestClassifier (class_weight = 'balanced',max_depth=10,n_estimators=100
clf_2v.fit(set2_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_2v.predict_proba(set2_train)[:
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_2v.predict_proba(set2_test)[:,1]

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



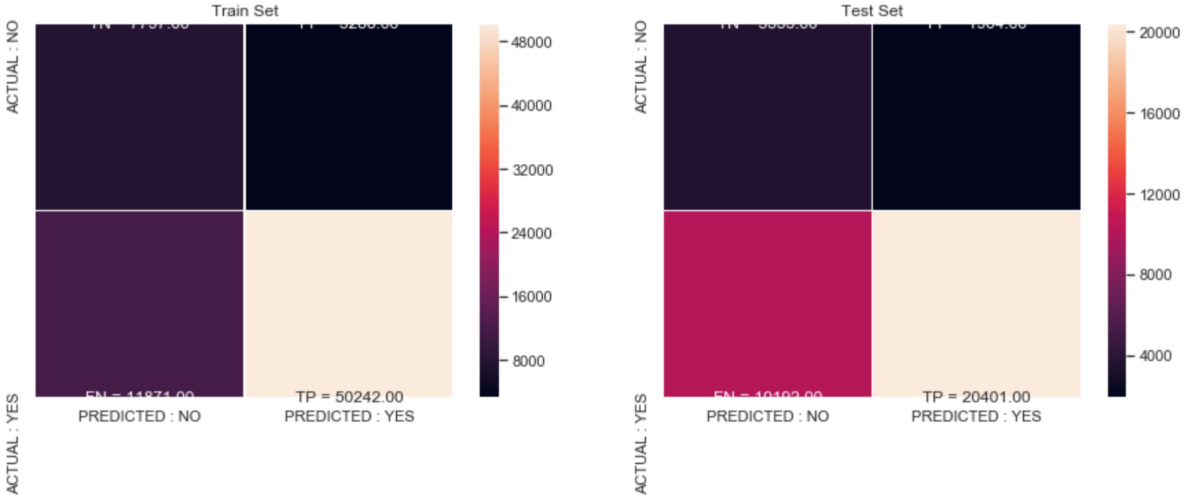
```
Wall time: 5min 45s
```

2.2.3 Confusion Matrix of set2_train , set2_test, TFIDF

In [255]:

```
clf_2v.fit(set2_train,y_train)
y_train_pred_2 = clf_2v.predict_proba(set2_train)[:,1]
y_test_pred_2 = clf_2v.predict_proba(set2_test)[:,1]

#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
conf_matr_df_train_2 = confusion_matrix(y_train, predict(y_train_pred_2, thresholds, tr
conf_matr_df_test_2 = confusion_matrix(y_test, predict(y_test_pred_2, thresholds, test_

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k

labels_test  = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k

sns.heatmap(conf_matr_df_train_2, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''

sns.heatmap(conf_matr_df_test_2, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.58 for threshold 0.5
the maximum value of tpr*(1-fpr) 0.43 for threshold 0.51
```



In [256]:

```
print("Confusion Matrix of set2_train of TFIDF:",conf_matr_df_train_2)
print("Confusion Matrix of set2_test of TFIDF:",conf_matr_df_test_2)
```

```
Confusion Matrix of set2_train of TFIDF: [[ 7797  3286]
 [11871 50242]]
Confusion Matrix of set2_test of TFIDF: [[ 3555  1904]
 [10192 20401]]
```

2.3 Applying Random Forest on AVG W2V, SET 3

In [224]:

```
# https://docs.scipy.org/doc/numpy/reference/generated/numpy.concatenate.html#numpy.con
# https://stackoverflow.com/questions/7922487/how-to-transform-numpy-matrix-or-array-to
# with the same np.concatenate function we are concatinating  matrix
# set3 = all categorical features + numarical features + essays_avg_w2v + project_title_
from scipy import sparse

set3_concat_train = np.concatenate((cat_0_train,cat_1_train, subcat_0_train,subcat_1_tr
                    teacher_prefix_0_train, teacher_prefix_1_train, proj_grade_0_train,
                    essays_avg_w2v_vectors_train,project_title_avg_w2v_vectors_train,pr
                    teacher_number_of_previously_posted_projects_normalizer_train),axis

set3_concat_test = np.concatenate((cat_0_test,cat_1_test, subcat_0_test,subcat_1_test,
                                teacher_prefix_0_test,teacher_prefix_1_test, proj_grade
                                essays_avg_w2v_vectors_test,project_title_avg_w2v_vecto
                                teacher_number_of_previously_posted_projects_normalizer

set3_train = sparse.csr_matrix(set3_concat_train)
set3_test  = sparse.csr_matrix(set3_concat_test)
print("Shape of set3_train Matrix :",set3_train.shape)
print("Shape of set3_test  Matrix :",set3_test.shape)
```

```
Shape of set3_train Matrix : (73196, 612)
Shape of set3_test  Matrix : (36052, 612)
```

2.3.1 Hyperparameter tuning to find best estimator : set3_RandomizedSearchCV

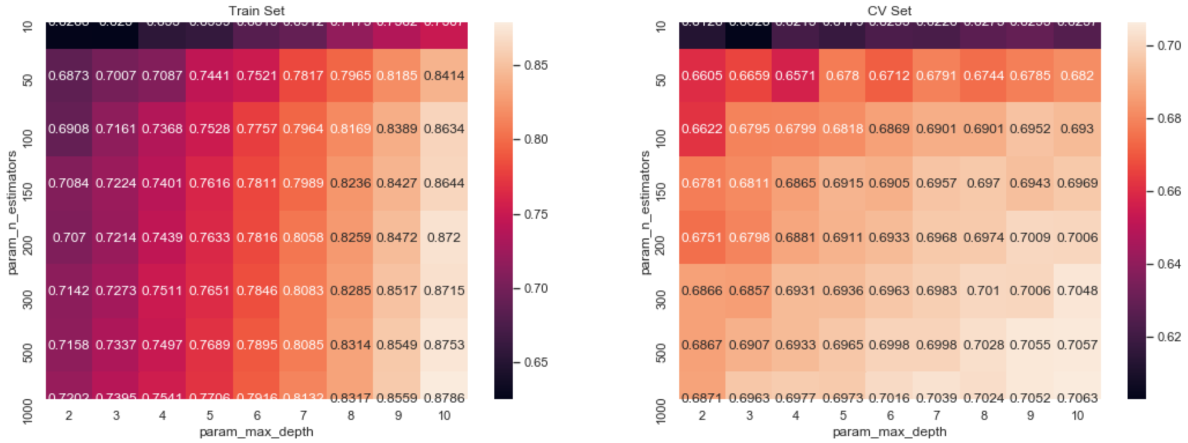
In [225]:

```
%%time
from sklearn.model_selection import RandomizedSearchCV
rfc3 = RandomForestClassifier(class_weight = 'balanced')
parameters = {'max_depth': [2,3,4,5,6,7,8,9,10], 'n_estimators': [10, 50, 100, 150, 200
clf3 = RandomizedSearchCV(rfc3, parameters, cv=3, scoring='roc_auc',return_train_score=
set3_fit = clf3.fit(set3_train, y_train)
```

```
Wall time: 2h 47min 43s
```

In [226]:

```
import seaborn as sns; sns.set()
max_scores3 = pd.DataFrame(clf3.cv_results_).groupby(['param_n_estimators', 'param_max_

fig, ax = plt.subplots(1,2, figsize=(18,6))
sns.heatmap(max_scores2.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores2.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 [227]:

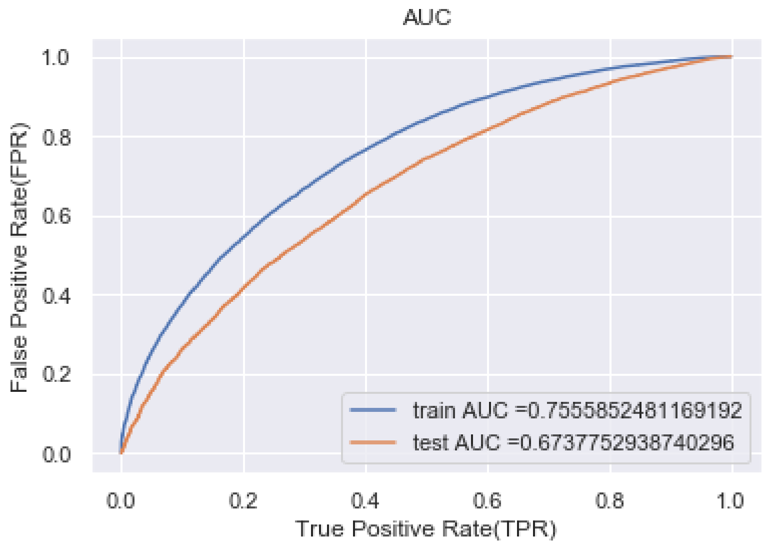
```
print(clf3.best_estimator_)
print(clf3.score(set3_train,y_train))
print(clf3.score(set3_test,y_test))
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=6, 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)
0.7561614130413284
0.6743581131032402
```

2.3.2 Train model using best estimator : set3

In [257]:

```
%%time
from sklearn.metrics import auc,roc_curve

clf_3v =RandomForestClassifier (class_weight = 'balanced',max_depth=6,n_estimators=1000
clf_3v.fit(set3_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_3v.predict_proba(set3_train)[:
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_3v.predict_proba(set3_test)[:,1]

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



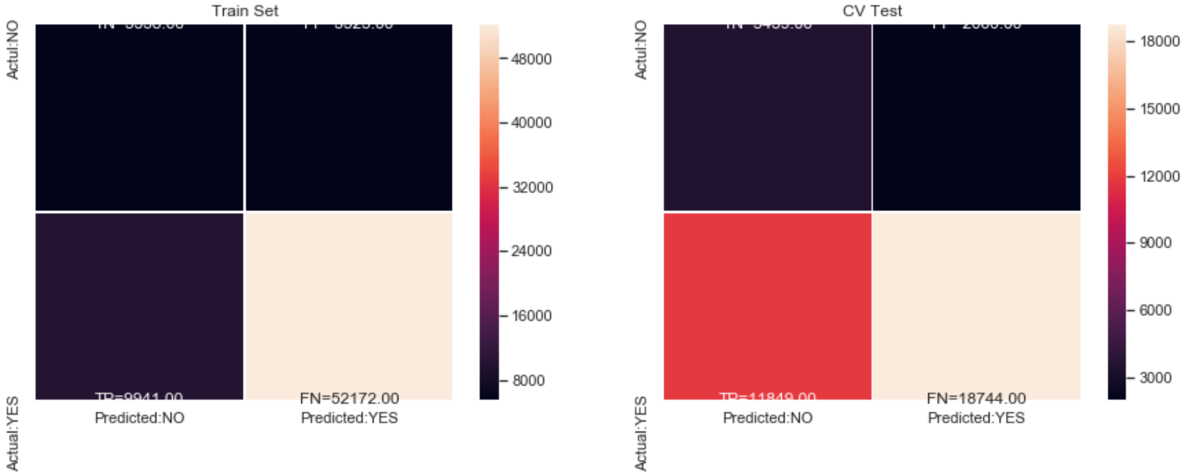
```
Wall time: 29min 4s
```

2.3.3 Confusion Matrix:: set3_train , set3_test

In [258]:

```
clf_3v.fit(set3_train,y_train)
y_train_pred_3 = clf_3v.predict_proba(set3_train)[:,1]
y_test_pred_3 = clf_3v.predict_proba(set3_test)[:,1]

conf_matr_df_train_3 = confusion_matrix(y_train,predict(y_train_pred_3,thresholds,train_
conf_matr_df_test_3  = confusion_matrix(y_test, predict(y_test_pred_3, thresholds, test_

key=(np.asarray([['TN','FP'],['TP','FN']]))
fig,ax=plt.subplots(1,2,figsize=(15,5))
labels_train = np.asarray(['{0}={1:.2f}'.format(key,value) for key, value in zip(key.fl
labels_test  = np.asarray(['{0}={1:.2f}'.format(key,value) for key ,value in zip(key.fl

sns.heatmap(conf_matr_df_train_3, linewidths=.5, xticklabels=['Predicted:NO','Predicted
sns.heatmap(conf_matr_df_test_3,  linewidths=.5, xticklabels=['Predicted:NO','Predicted
ax[0].set_title('Train Set')
ax[1].set_title("CV Test")
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.47 for threshold 0.48
the maximum value of tpr*(1-fpr) 0.39 for threshold 0.51
```



In [259]:

```
conf_matr_df_test_3
```

Out[259]:

```
array([[ 3459,  2000],
       [11849, 18744]], dtype=int64)
```

2.4 Apply Random Forest on set4_TFIDF w2v

In [228]:

```
# set_4 = encoded numarical + categorical + project_title_tfidf_w2v_vectors +essays_tfi
set4_concat_train = np.concatenate((cat_0_train,cat_1_train, subcat_0_train,subcat_1_tr
                    teacher_prefix_0_train, teacher_prefix_1_train, proj_grade_0_train,
                    preprocessed_essays_train_tfidf_w2v_vectors,preprocessed_project_ti
                    price_normalizer_train,teacher_number_of_previously_posted_projects_

set4_concat_test = np.concatenate((cat_0_test,cat_1_test, subcat_0_test,subcat_1_test, 
                                teacher_prefix_0_test,teacher_prefix_1_test, proj_grade_
                        preprocessed_essays_test_tfidf_w2v_vectors,preprocessed_project_
                                price_normalizer_test,teacher_number_of_previously_post

set4_train = sparse.csr_matrix(set4_concat_train)
set4_test  = sparse.csr_matrix(set4_concat_test)
print("Shape of set4_train Matrix :",set4_train.shape)
print("Shape of set4_test  Matrix :",set4_test.shape)
```

```
Shape of set4_train Matrix : (73196, 612)
Shape of set4_test  Matrix : (36052, 612)
```

2.4.1 Hyperparameter Tuning to find the best estimator : set4_RandomizedSearchCV

In [229]:

```
%%time
rfc4 = RandomForestClassifier(class_weight='balanced')
parameters = {'max_depth': [2,3,4,5,6,7,8,9,10], 'n_estimators': [10, 50, 100, 150, 200
clf4 = RandomizedSearchCV(rfc4,parameters, cv=3, scoring='roc_auc', return_train_score=
clf4.fit(set4_train,y_train)
```

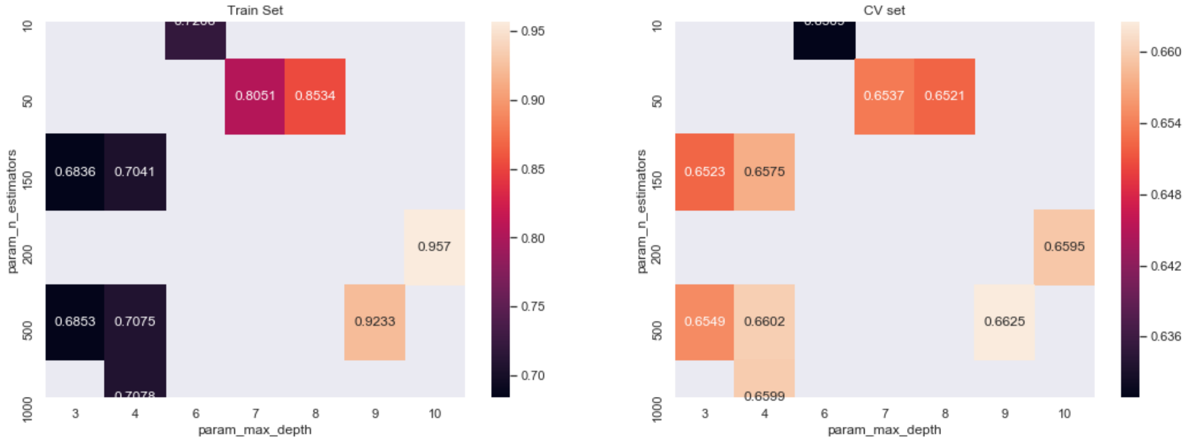
```
Wall time: 1h 43min 41s
```

Out[229]:

```
RandomizedSearchCV(cv=3, error_score='raise-deprecating',
                   estimator=RandomForestClassifier(bootstrap=True,
                                                    class_weight='balance
d',
                                                    criterion='gini',
                                                    max_depth=None,
                                                    max_features='auto',
                                                    max_leaf_nodes=None,
                                                    min_impurity_decrease=
0.0,
                                                    min_impurity_split=Non
e,
                                                    min_samples_leaf=1,
                                                    min_samples_split=2,
                                                    min_weight_fraction_le
af=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_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='roc_auc', verbose=0)
```

In [230]:

```
import seaborn as sns; sns.set()
max_score_4 = pd.DataFrame(clf4.cv_results_).groupby(['param_n_estimators','param_max_d
    'mean_test_score','mean_train_score']]

fig,ax = plt.subplots(1,2,figsize=(18,6))
sns.heatmap(max_score_4.mean_train_score, annot=True, fmt='.4g',ax=ax[0])
sns.heatmap(max_score_4.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 [231]:

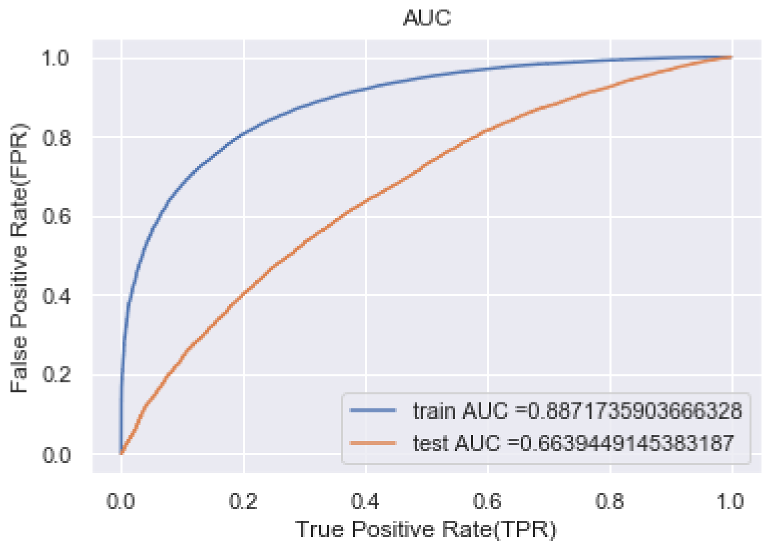
```
# best estimator and best parameters
print(clf4.best_estimator_)
print(clf4.score(set4_train,y_train))
print(clf4.score(set4_test, y_test))
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       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=500, n_jobs=None, oob_score=False,
                       random_state=None, verbose=0, warm_start=False)
0.8863872310193223
0.6630883316416796
```

2.4.2 Train model using the best hyper-parameter value :set4 TFIDF W2V

In [260]:

```
%%time
from sklearn.metrics import auc,roc_curve

clf_4m =RandomForestClassifier (class_weight = 'balanced',max_depth=9,n_estimators=500)
clf_4m.fit(set4_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs
train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_4m.predict_proba(set4_train)[:
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_4m.predict_proba(set4_test)[:,1]

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



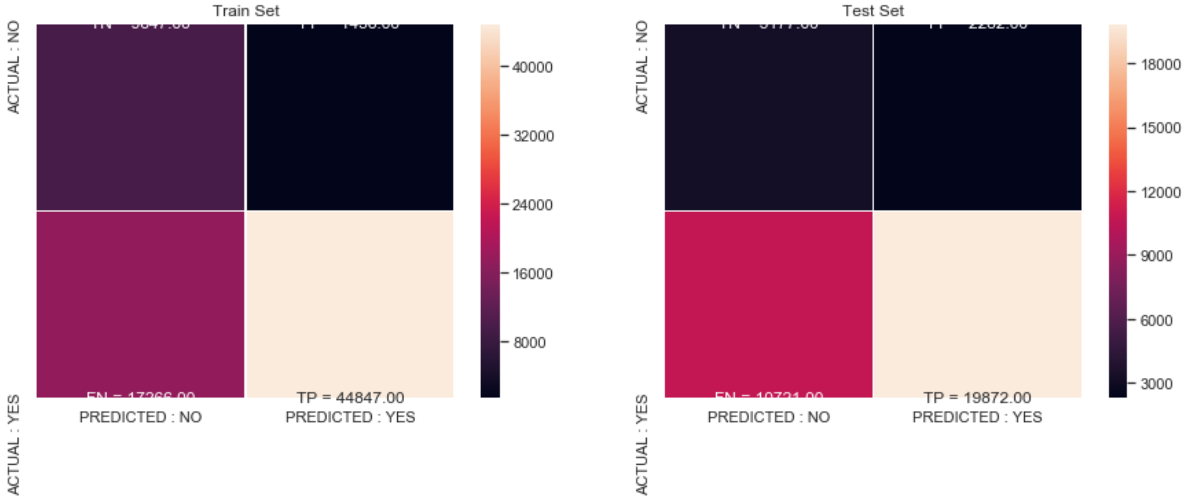
```
Wall time: 31min 31s
```

2.4.3 Confustion Matrix set4_train and set4_test

In [261]:

```
clf_4m.fit(set4_train,y_train)
y_train_pred_4 = clf_4m.predict_proba(set4_train)[:,1]
y_test_pred_4 = clf_4m.predict_proba(set4_test)[:,1]
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
conf_matr_df_train_4 = confusion_matrix(y_train, predict(y_train_pred_4, thresholds, tr
conf_matr_df_test_4 = confusion_matrix(y_test, predict(y_test_pred_4, thresholds, test_

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k
labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key

sns.heatmap(conf_matr_df_train_4, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''
sns.heatmap(conf_matr_df_test_4, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',
ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.65 for threshold 0.52
the maximum value of tpr*(1-fpr) 0.38 for threshold 0.53
```



In [262]:

```
conf_matr_df_test_4
```

Out[262]:

```
array([[ 3177,  2282],
       [10721, 19872]], dtype=int64)
```

2.5 Getting top 5k features using feature_importances_with high AUC set - Experiment`

In [232]:

```
def selectKImportance(model, X, k=5):
    return X[:,model.best_estimator_.feature_importances_.argsort()[::-1][:k]]
```

In [233]:

```
# for set3
set5_train = selectKImportance(clf2, set2_train,5000)
set5_test = selectKImportance(clf2, set2_test, 5000)
print(set5_train.shape)
print(set5_test.shape)
```

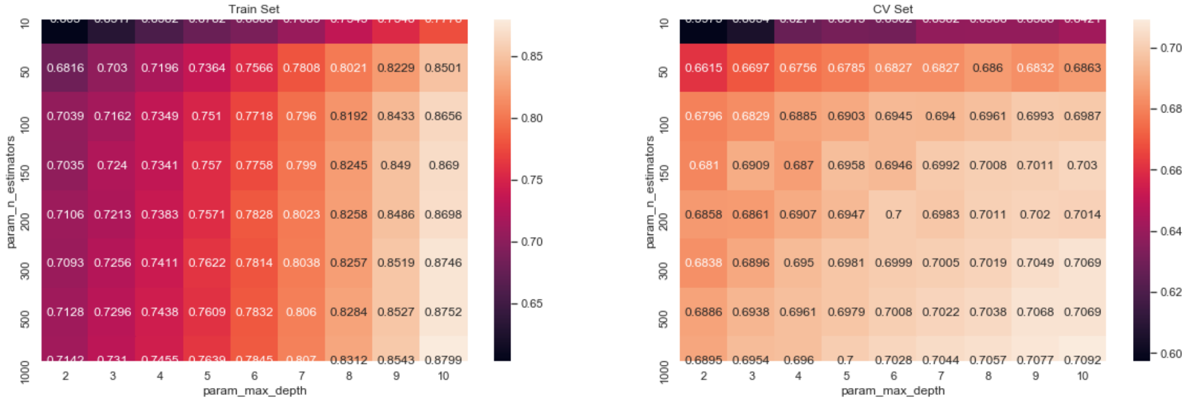
```
(73196, 5000)
(36052, 5000)
```

2.5.1 Apply Random Forest on Important features of set2

Hyperparameter Tuning to find the best estimator

In [234]:

```
%%time
rfc5= RandomForestClassifier(class_weight = 'balanced')
parameters = {'max_depth': [2,3,4,5,6,7,8,9,10], 'n_estimators': [10, 50, 100, 150, 200
clf5 = GridSearchCV(rfc5, parameters, cv=3, scoring='roc_auc',return_train_score=True)
set5_fit= clf5.fit(set5_train, y_train)
import seaborn as sns; sns.set()
max_scores5 = pd.DataFrame(clf5.cv_results_).groupby(['param_n_estimators', 'param_max_
    'mean_test_score', 'mean_train_score']]

fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores5.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores5.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



```
Wall time: 2h 58min 35s
```

In [235]:

```
#Best Estimator and Best tune parameters
print(clf5.best_estimator_)
#Mean cross-validated score of the best_estimator
print(clf5.score(set5_train,y_train))
print(clf5.score(set5_test,y_test))
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
                       criterion='gini', max_depth=10, max_features='aut
o',
                       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)
0.8475426748499071
0.7152372490412643
```

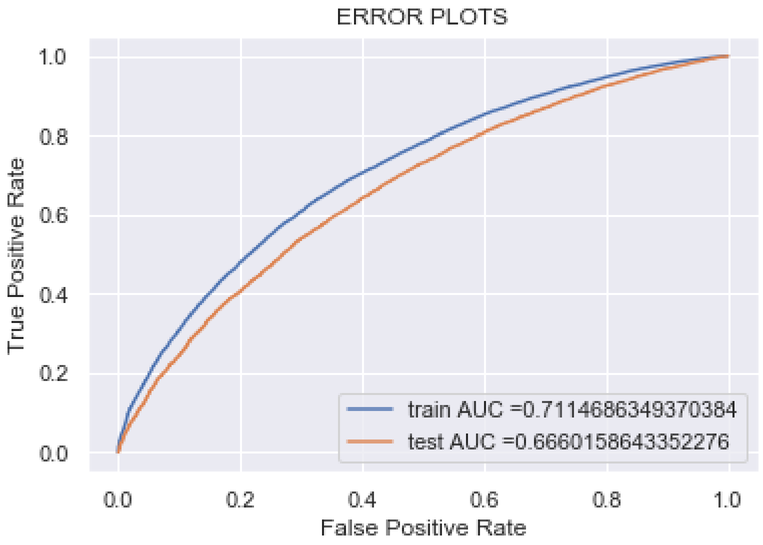
In [236]:

```
# Best tune parameters
best_tune_parameters=[{'max_depth': [10], 'min_samples_split':[1000] } ]
```

2.4.2 Train model using the best estimator :

In [263]:

```
clf5= GridSearchCV( RandomForestClassifier(class_weight = 'balanced'),best_tune_paramet

clfV5=RandomForestClassifier (class_weight = 'balanced',max_depth=10,n_estimators=1000)

clf5.fit(set5_train, y_train)
# for visulation
clfV5.fit(set5_train, y_train)
#https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.ht
#sklearn.linear_model.SGDClassifier.decision_function
y_train_pred5 = clf5.predict_proba(set5_train) [:,1]
y_test_pred5 = clf5.predict_proba(set5_test) [:,1]

train_fpr, train_tpr, thresholds = roc_curve(y_train, y_train_pred5)
test_fpr, test_tpr, thresholds = roc_curve(y_test, y_test_pred5)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```

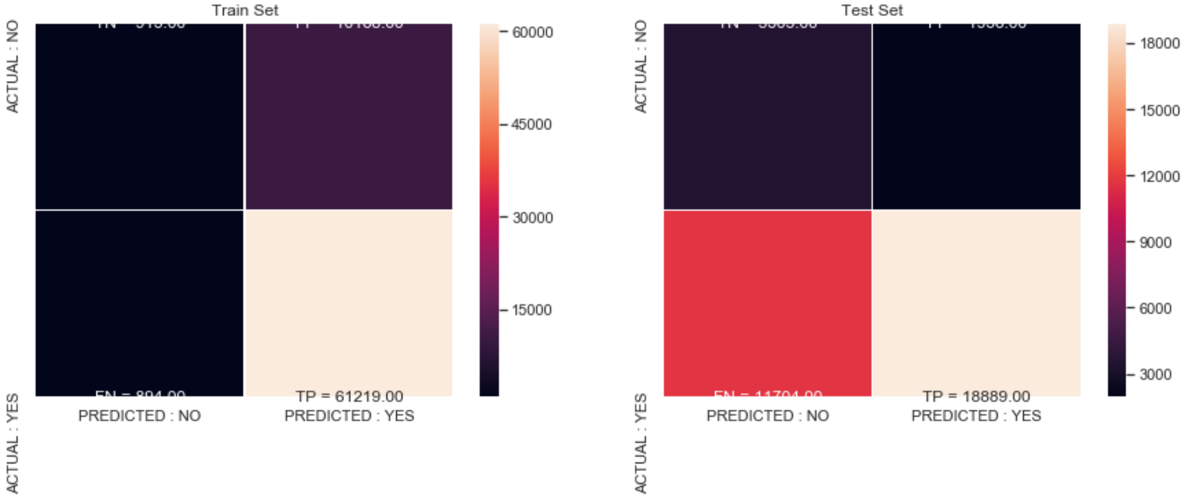


2.5.3 Confusion Matrix ::

In [264]:

```
clf5.fit(set5_train,y_train)
y_train_pred_5 = clf5.predict_proba(set5_train)[:,1]
y_test_pred_5  = clf5.predict_proba(set5_test)[:,1]


#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
conf_matr_df_train_5 = confusion_matrix(y_train, predict(y_train_pred_5, thresholds, tr
conf_matr_df_test_5  = confusion_matrix(y_test, predict(y_test_pred_5, thresholds, test_

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k

labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(ke

sns.heatmap(conf_matr_df_train_5, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''

sns.heatmap(conf_matr_df_test_5, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.43 for threshold 0.42
the maximum value of tpr*(1-fpr) 0.39 for threshold 0.51
```



In [265]:

```
conf_matr_df_test_5
```

Out[265]:

```
array([[ 3503,  1956],
       [11704, 18889]], dtype=int64)
```

In [515]:

```
pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\programdata\anaconda3\lib\sit
e-packages (0.90)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.16.5)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-
packages (from xgboost) (1.3.1)
Note: you may need to restart the kernel to use updated packages.
```

2.6 Apply XGBoost on set1_RandomizedSearchCV

In [237]:

```
%%time
#https://xgboost.readthedocs.io/en/latest/python/python_api.html#module-xgboost.sklearn

from xgboost import XGBClassifier

xgbc1 = XGBClassifier(class_weight = 'balanced')
parameters = {'max_depth': [2,3,4,5,6,7,8,9,10], 'n_estimators': [10, 50, 100, 150, 200
clf_x1 = RandomizedSearchCV(xgbc1,parameters, cv=3, scoring='roc_auc', return_train_sco
clf_x1.fit(set1_train,y_train)
```

```
Wall time: 3h 2min 58s
```

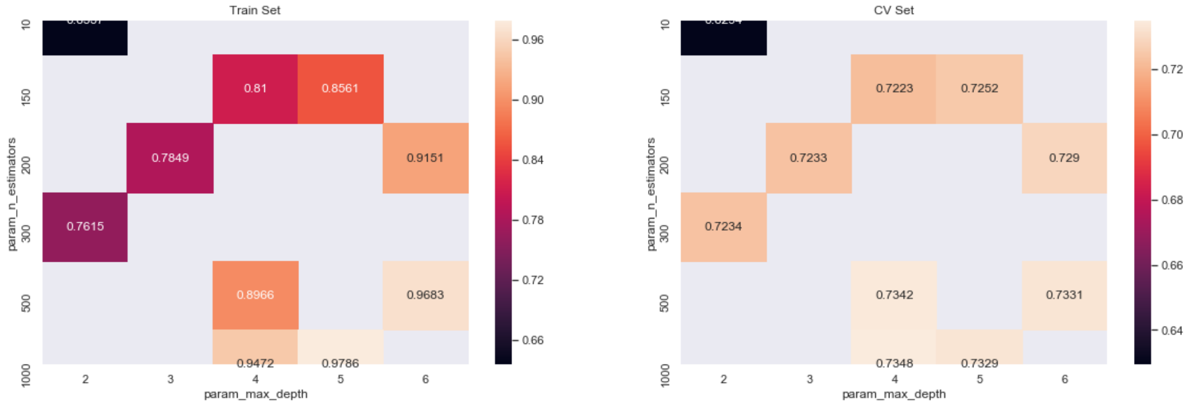
Out[237]:

```
RandomizedSearchCV(cv=3, error_score='raise-deprecating',
                   estimator=XGBClassifier(base_score=0.5, booster='gbtre
e',
                                           class_weight='balanced',
                                           colsample_bylevel=1,
                                           colsample_bynode=1,
                                           colsample_bytree=1, gamma=0,
                                           learning_rate=0.1, max_delta_st
ep=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, subsamp
le=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': [10, 50, 100, 150,
200,
                                                         300, 500, 1000]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='roc_auc', verbose=0)
```

2.6.1 Hyperparameter Tuning to find the best estimator : set1_bow_XGBoost

In [238]:

```
max_scores_x1 = pd.DataFrame(clf_x1.cv_results_).groupby(['param_n_estimators', 'param_

fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores_x1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores_x1.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 [239]:

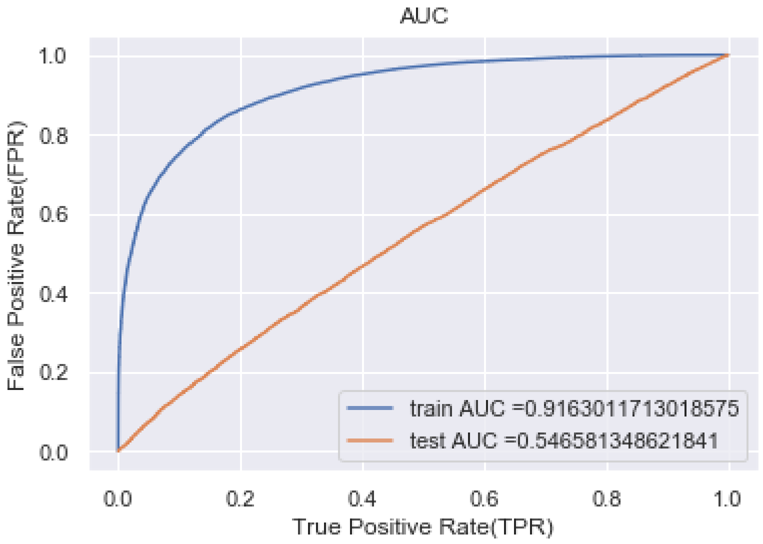
```
print(clf_x1.best_estimator_)
print(clf_x1.score(set1_train,y_train))
print(clf_x1.score(set1_test, y_test))
```

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=4,
              min_child_weight=1, missing=None, n_estimators=1000, 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)
0.9163011713018575
0.546581348621841
```

2.6.2 Train model using the best estimator : set1_XGBoost

In [266]:

```
clf_x1v = XGBClassifier(class_weight = 'balanced',max_depth=4,n_estimators=1000)
clf_x1v.fit(set1_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_x1v.predict_proba(set1_train)[
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_x1v.predict_proba(set1_test)[:,1

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

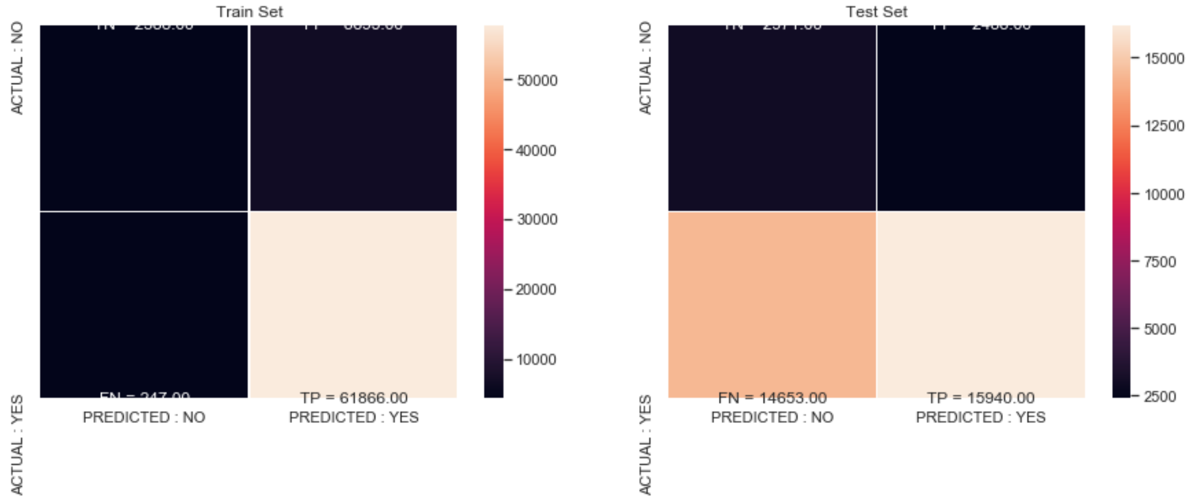


2.6.3 Confusion Matrix :: set1_train , set1_test , XGBoost

In [267]:

```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
clf_x1v.fit(set1_train,y_train)
y_train_pred_x1 = clf_x1v.predict_proba(set1_train)[:,1]
y_test_pred_x1 = clf_x1v.predict_proba(set1_test)[:,1]

conf_matr_df_train_x1 = confusion_matrix(y_train, predict(y_train_pred_x1, thresholds, 
conf_matr_df_test_x1 = confusion_matrix(y_test, predict(y_test_pred_x1, thresholds, tes

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k

labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(ke

sns.heatmap(conf_matr_df_train_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''

sns.heatmap(conf_matr_df_test_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.7 for threshold 0.54
the maximum value of tpr*(1-fpr) 0.29 for threshold 0.6
```



2.7 Applying XGBoost on TFIDF set2:: RandomizedSearchCV

In [240]:

```
%%time
xgbc2 = XGBClassifier(class_weight = 'balanced')
parameters = {'max_depth': [2,3,4,5,6,7,8,9,10], 'n_estimators': [10, 50, 100, 150, 200
clf_x2 = RandomizedSearchCV(xgbc2,parameters, cv=3, scoring='roc_auc', return_train_sco
clf_x2.fit(set2_train,y_train)
```

```
Wall time: 5h 1min 34s
```

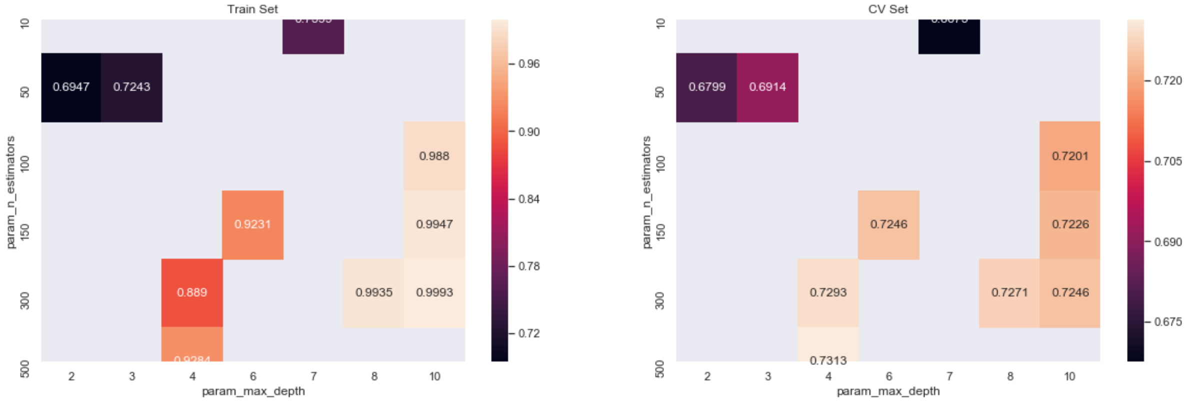
Out[240]:

```
RandomizedSearchCV(cv=3, error_score='raise-deprecating',
                   estimator=XGBClassifier(base_score=0.5, booster='gbtre
e',
                                           class_weight='balanced',
                                           colsample_bylevel=1,
                                           colsample_bynode=1,
                                           colsample_bytree=1, gamma=0,
                                           learning_rate=0.1, max_delta_st
ep=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, subsamp
le=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': [10, 50, 100, 150,
200,
                                                         300, 500, 1000]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='roc_auc', verbose=0)
```

2.7.1 Hyperparameter Tuning to find the best estimator : set2_tfidf, XGBoost

In [241]:

```
max_scores_x2 = pd.DataFrame(clf_x2.cv_results_).groupby(['param_n_estimators', 'param_
    ['mean_test_score', 'mean_train_score']]

fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores_x2.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores_x2.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 [242]:

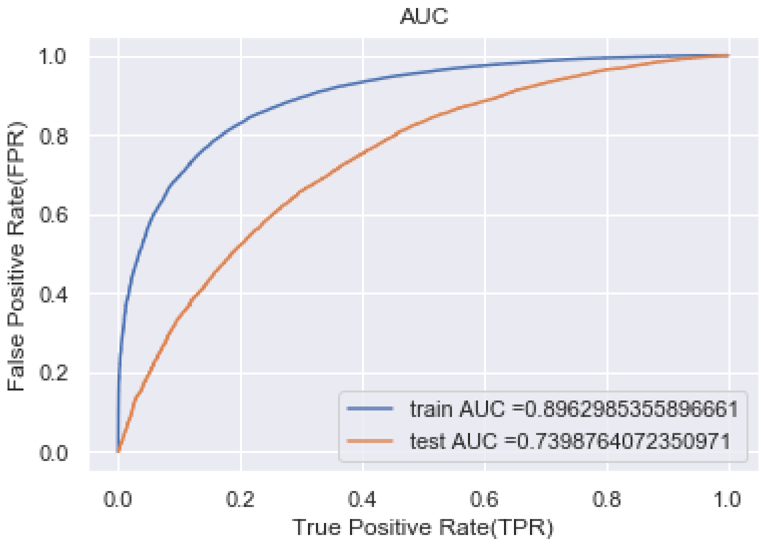
```
print(clf_x2.best_estimator_)
print(clf_x2.score(set2_train,y_train))
print(clf_x2.score(set2_test,y_test))
```

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=4,
              min_child_weight=1, missing=None, n_estimators=500, 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)
0.8962985355896661
0.7398764072350971
```

2.7.2 Train model using the best estimator : set2

In [268]:

```
clf_x2v = XGBClassifier(class_weight = 'balanced',max_depth=4,n_estimators=500)
clf_x2v.fit(set2_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of th
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_x2v.predict_proba(set2_train)[
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_x2v.predict_proba(set2_test)[:,1

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

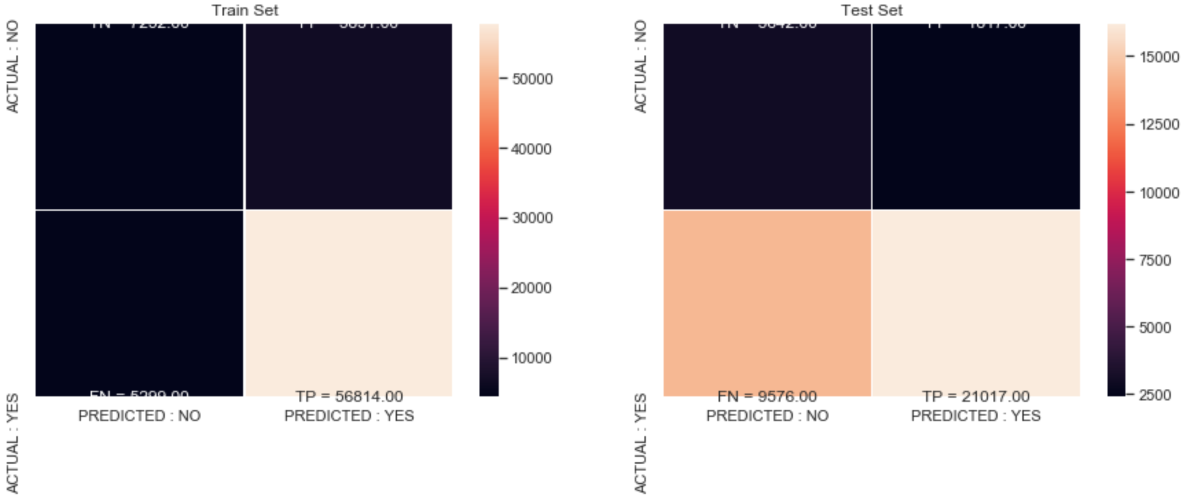


2.7.3 Confusion Matrix : set2_train, set2_test - XGbbost

In [269]:

```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
clf_x2v.fit(set2_train,y_train)
y_train_pred_x2 = clf_x2v.predict_proba(set2_train)[:,1]
y_test_pred_x2 = clf_x2v.predict_proba(set2_test)[:,1]

conf_matr_df_train_x2 = confusion_matrix(y_train, predict(y_train_pred_x2, thresholds, t
conf_matr_df_test_x2 = confusion_matrix(y_test, predict(y_test_pred_x2, thresholds, tes

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k

labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key

sns.heatmap(conf_matr_df_train_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''

sns.heatmap(conf_matr_df_test_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.66 for threshold 0.75
the maximum value of tpr*(1-fpr) 0.46 for threshold 0.84
```



2.8 Apply XGBoost Classifier on Avg W2V set3 : RandomizedSearchCV

In [243]:

```
%%time
xgbc3 = XGBClassifier(class_weight = 'balanced')
parameters = {'max_depth': [2,3,4,5,6,7,8,9,10], 'n_estimators': [10, 50, 100, 150, 200
clf_x3 = RandomizedSearchCV(xgbc3,parameters, cv=3, scoring='roc_auc', return_train_sco
clf_x3.fit(set3_train,y_train)
```

```
Wall time: 20h 38min 19s
```

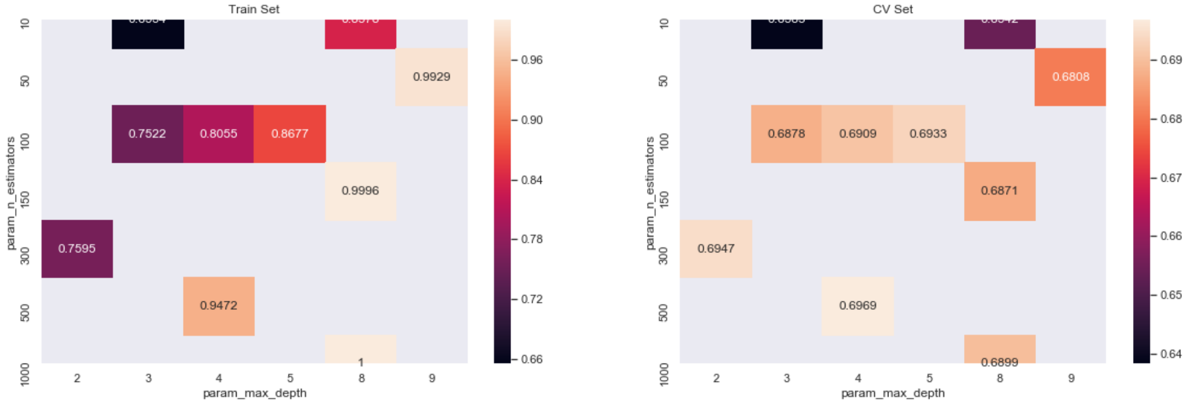
Out[243]:

```
RandomizedSearchCV(cv=3, error_score='raise-deprecating',
                   estimator=XGBClassifier(base_score=0.5, booster='gbtre
e',
                                           class_weight='balanced',
                                           colsample_bylevel=1,
                                           colsample_bynode=1,
                                           colsample_bytree=1, gamma=0,
                                           learning_rate=0.1, max_delta_st
ep=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, subsamp
le=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': [10, 50, 100, 150,
200,
                                                         300, 500, 1000]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='roc_auc', verbose=0)
```

2.8.1 Hyperparameter Tuning to find the best estimator : set3_XGBoost

In [244]:

```
%%time
max_scores_x3 = pd.DataFrame(clf_x3.cv_results_).groupby(['param_n_estimators', 'param_
    ['mean_test_score', 'mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores_x3.mean_train_score,annot=True,fmt='.4g',ax=ax[0])
sns.heatmap(max_scores_x3.mean_test_score, annot=True,fmt='.4g', ax= ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



```
Wall time: 24.6 s
Parser   : 1.01 s
```

In [245]:

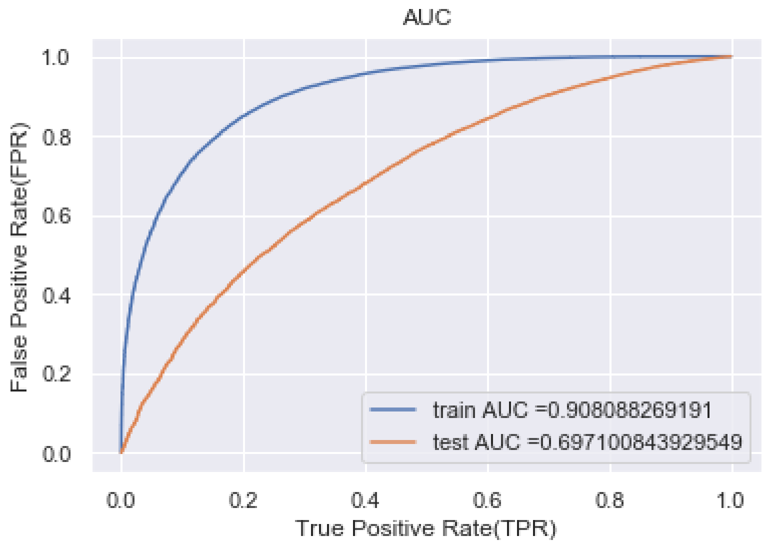
```
print(clf_x3.best_estimator_)
print(clf_x3.score(set3_train,y_train))
print(clf_x3.score(set3_test,y_test))
```

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balanced',
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
              gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=4,
              min_child_weight=1, missing=None, n_estimators=500, 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)
0.908088269191
0.697100843929549
```

2.8.2 Train model using the best estimator: set3_XGboost

In [270]:

```
%%time
clf_x3v = XGBClassifier(class_weight = 'balanced',max_depth=4,n_estimators=500)
clf_x3v.fit(set3_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_x3v.predict_proba(set3_train)[
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_x3v.predict_proba(set3_test)[:,1

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



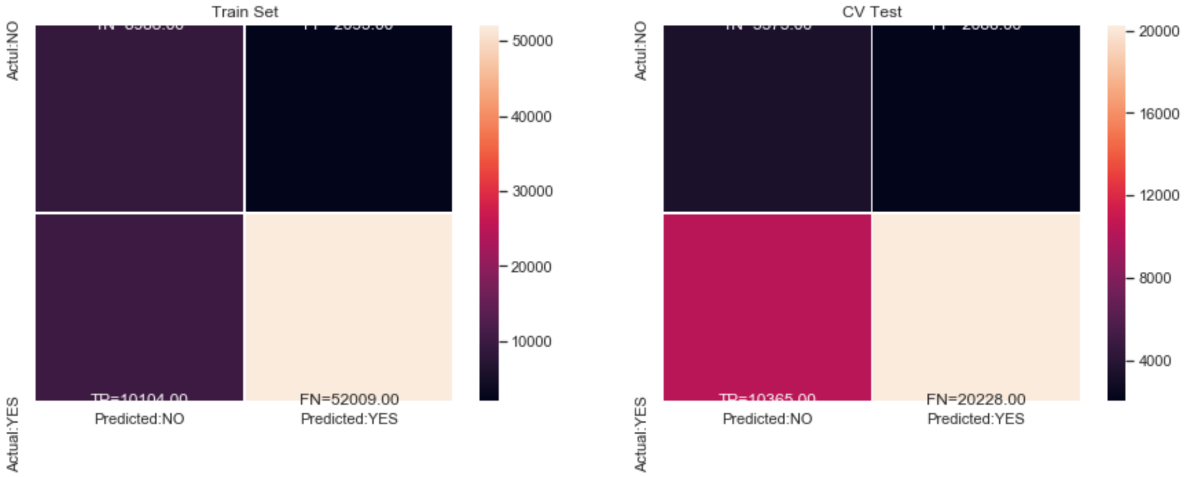
```
Wall time: 1h 2min 21s
```

2.8.3 Confusion Matrix : set3_train, set3_test - XGBoost

In [271]:

```
%%time
clf_x3v.fit(set3_train,y_train)
y_train_pred_x3 = clf_x3v.predict_proba(set3_train)[:,1]
y_test_pred_x3 = clf_x3v.predict_proba(set3_test)[:,1]

conf_matr_df_train_x3 = confusion_matrix(y_train,predict(y_train_pred_x3,thresholds,tra
conf_matr_df_test_x3  = confusion_matrix(y_test, predict(y_test_pred_x3, thresholds, te

key=(np.asarray([['TN','FP'],['TP','FN']]))
fig,ax=plt.subplots(1,2,figsize=(15,5))
labels_train = np.asarray(['{0}={1:.2f}'.format(key,value) for key, value in zip(key.fl
labels_test  = np.asarray(['{0}={1:.2f}'.format(key,value) for key ,value in zip(key.fl

sns.heatmap(conf_matr_df_train_x3, linewidths=.5, xticklabels=['Predicted:NO','Predicte
sns.heatmap(conf_matr_df_test_x3,  linewidths=.5, xticklabels=['Predicted:NO','Predicte
ax[0].set_title('Train Set')
ax[1].set_title("CV Test")
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.68 for threshold 0.81
the maximum value of tpr*(1-fpr) 0.41 for threshold 0.84
```



```
Wall time: 1h 1min 48s
Compiler : 301 ms
Parser   : 1.38 s
```

2.9 Apply XGBoost Classifier on set4 TFIDF W2V: RandomizedSearchCV

In [ ]:

```
%%time
xgbc4 = XGBClassifier(class_weight = 'balanced')
parameters = {'max_depth': [2,3,4,5,6,7,8,9,10], 'n_estimators': [10, 50, 100, 150, 200
clf_x4 = RandomizedSearchCV(xgbc4,parameters, cv=3, scoring='roc_auc', return_train_sco
clf_x4.fit(set4_train,y_train)
```

2.9.1 Hyperparameter Tuning to find the best estmator : set4_XGBoost

In [247]:

```
%%time
max_scores_x4 = pd.DataFrame(clf_x4.cv_results_).groupby(['param_n_estimators','param_m
    'mean_test_score','mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores_x4.mean_train_score,annot=True,fmt='.4g',ax=ax[0])
sns.heatmap(max_scores_x4.mean_test_score, annot=True,fmt='.4g', ax= ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<timed exec> in <module>

AttributeError: 'RandomizedSearchCV' object has no attribute 'cv_results_'
```

In [248]:

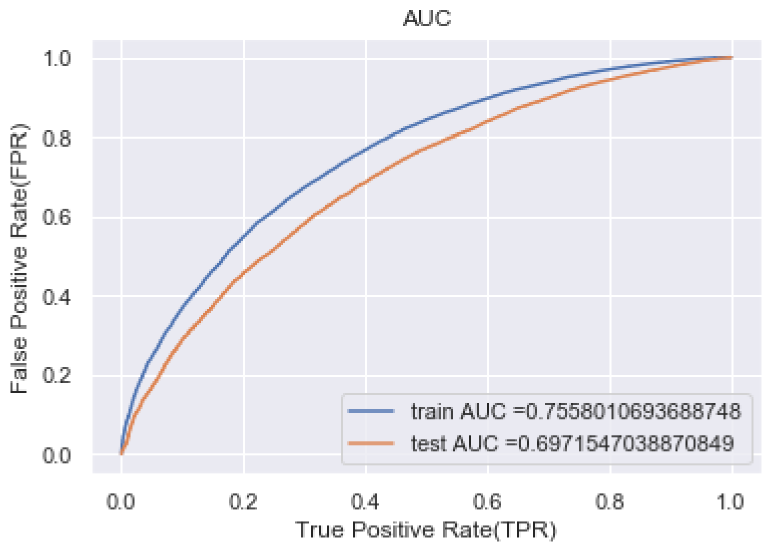
```
print(clf_x4.best_estimator_)
print(clf_x4.score(set4_train,y_train))
print(clf_x4.score(set4_test,y_test))
```

```
---------------------------------------------------------------------------
-
AttributeError                            Traceback (most recent call las
t)
<ipython-input-248-e770aec0fc5d> in <module>
----> 1 print(clf_x4.best_estimator_)
      2 print(clf_x4.score(set4_train,y_train))
      3 print(clf_x4.score(set4_test,y_test))

AttributeError: 'RandomizedSearchCV' object has no attribute 'best_estimat
or_'
```

2.9.2 Train model using the best estimator : set4

In [534]:

```
%%time
clf_x4v = XGBClassifier(class_weight = 'balanced',max_depth=2,n_estimators=500)
clf_x4v.fit(set4_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_x4v.predict_proba(set4_train)[
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_x4v.predict_proba(set4_test)[:,1

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



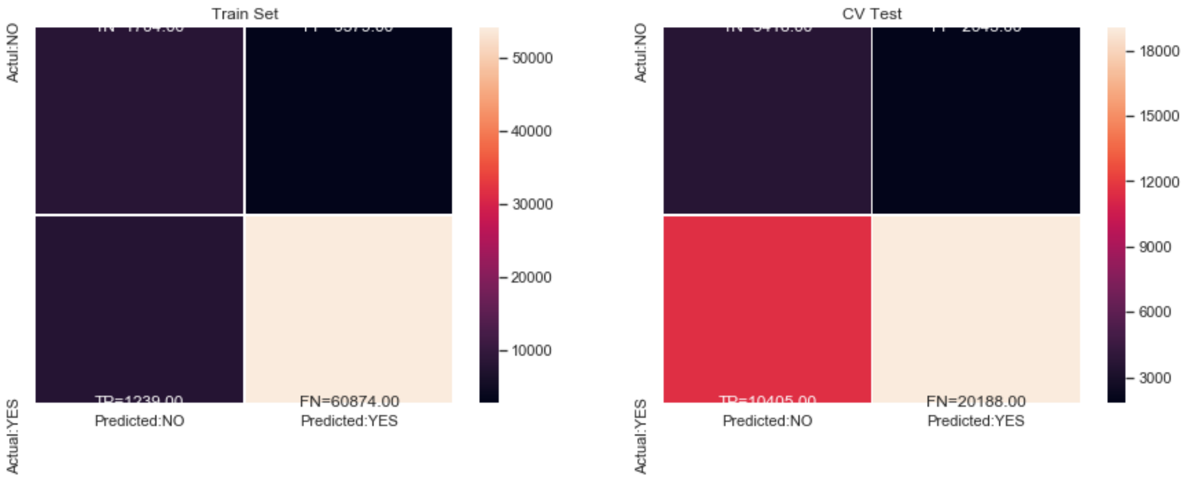
```
Wall time: 20min 23s
```

2.9.3 Confusion Matrix : set4_train, set4_test - XGBoost

In [535]:

```
clf_x4v.fit(set4_train,y_train)
y_train_pred_x4 = clf_x4v.predict_proba(set4_train)[:,1]
y_test_pred_x4 = clf_x4v.predict_proba(set4_test)[:,1]

conf_matr_df_train_x4 = confusion_matrix(y_train,predict(y_train_pred_x4,thresholds,tra
conf_matr_df_test_x4  = confusion_matrix(y_test, predict(y_test_pred_x4, thresholds, te

key=(np.asarray([['TN','FP'],['TP','FN']]))
fig,ax=plt.subplots(1,2,figsize=(15,5))
labels_train = np.asarray(['{0}={1:.2f}'.format(key,value) for key, value in zip(key.fl
labels_test  = np.asarray(['{0}={1:.2f}'.format(key,value) for key ,value in zip(key.fl

sns.heatmap(conf_matr_df_train_x3, linewidths=.5, xticklabels=['Predicted:NO','Predicte
sns.heatmap(conf_matr_df_test_x3,  linewidths=.5, xticklabels=['Predicted:NO','Predicte
ax[0].set_title('Train Set')
ax[1].set_title("CV Test")
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.47 for threshold 0.67
the maximum value of tpr*(1-fpr) 0.42 for threshold 0.82
```



2.10. Observation::

1. XGBOOST Model is performing better than Random Forest model, but XGBoost is taking to much time to run
2. From all the sets, TFIDF is working fairly well having AUC score of 0.74 for XGBOOST is the highest

3.Conclusion ::

In [2]:

```
# Link : http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
p = PrettyTable()

p.field_names = ["Vectorizer", "Model", "Best_param_max_depth","Best_param_min_samples_

p.add_row(["BOW-set1","Random Forest",10,1000, 0.57])
p.add_row(["BOW-set1","XGBoost",5,1000,0.56])
p.add_row(["TFIDF-set2","Random Forest",10,1000, 0.71])
p.add_row(["TFIDF-set2","XGBoost",3,1000,0.74])
p.add_row(["AVG W2V-set3","Random Forest",5,1000, 0.67])
p.add_row(["AVG W2V-set3","XGBoost",5,200,0.70])
p.add_row(["TFIDF W2V-set4","Random Forest",7,300, 0.66])
p.add_row(["TFIDF W2V-set4","XGBoost",2,500,0.69])
p.add_row(["Top 5K Points TFIDF-Set5-Experiment","RF",10, 500, 0.66])
print(p)
```

```
+-------------------------------------+---------------+-------------------
---+------------------------------+----------+
|              Vectorizer             |     Model     | Best_param_max_dep
th | Best_param_min_samples_split | Test AUC |
+-------------------------------------+---------------+-------------------
---+------------------------------+----------+
|               BOW-set1              | Random Forest |          10
|             1000             |   0.57   |
|               BOW-set1              |    XGBoost    |          5
|             1000             |   0.56   |
|              TFIDF-set2             | Random Forest |          10
|             1000             |   0.71   |
|              TFIDF-set2             |    XGBoost    |          3
|             1000             |   0.74   |
|             AVG W2V-set3            | Random Forest |          5
|             1000             |   0.67   |
|             AVG W2V-set3            |    XGBoost    |          5
|             200              |   0.7    |
|            TFIDF W2V-set4           | Random Forest |          7
|             300              |   0.66   |
|            TFIDF W2V-set4           |    XGBoost    |          2
|             500              |   0.69   |
| Top 5K Points TFIDF-Set5-Experiment |       RF      |          10
|             500              |   0.66   |
+-------------------------------------+---------------+-------------------
---+------------------------------+----------+
```

Thank You.

Sign Off RAMESH BATTU (https://www.linkedin.com/in/rameshbattuai/)