DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy`

Feature	Description
	• `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

your neighborhood, and your school are all helpful.

- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('train_data.csv',nrows=50000)
resource_data = pd.read_csv('resources.csv')
```

In [3]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

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

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

Out[4]:

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

In [5]:

```
y = project_data['project_is_approved'].values
#project_data.drop(['project_is_approved'], axis=1, inplace=True)
project_data.head(1)
```

Out[5]:

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

In [6]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
project_data.head(5)
```

Out[6]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Gra
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Gra

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Gra

In [7]:

```
X=project_data
X.head(5)
```

Out[7]:

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

1.2 preprocessing of `project_subject_categories`

In [8]:

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

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

project_data['clean_categories'] = cat_list
```

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [9]:

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

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

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

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]))
```

1.4 preprocessing of `project_grade_category`

In [10]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for j in catogories:
    temp = ""
    j=j.replace(' ','_')
    j = j.replace('-','To')
    temp+=j
    cat_list.append(temp)

project_data['project_grade_category'] = cat_list
```

1.5 Text preprocessing

In [11]:

```
# 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 [12]:

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

Out[12]:

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

In [13]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
```

```
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier that exists in schools for a child who can't sit
still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of
desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work
hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in
Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq
ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage
r learners; they are like sponges, absorbing all the information and experiences and keep on wanti
ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin
g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to
be one of a themed nautical environment. Creating a classroom environment is very important in the
success in each and every child's education. The nautical photo props will be used with each child
as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic
tures of each child with them, have them developed, and then hung in our classroom ready for their
first day of 4th grade.  This kind gesture will set the tone before even the first day of school!
The nautical thank you cards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of
my own pocket on resources to get our classroom ready. Please consider helping with this project t
o make our new school year a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out
for my students. I teach in a Title I school where most of the students receive free or reduced pr
ice lunch.  Despite their disabilities and limitations, my students love coming to school and come
eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to gr
oove and move as you were in a meeting? This is how my kids feel all the time. The want to be able
to move as they learn or so they say.Wobble chairs are the answer and I love then because they dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids don't want to sit and do worksheets. They want to learn to count by ju
mping and playing. Physical engagement is the key to our success. The number toss and color and sh
ape mats can make that happen. My students will forget they are doing work and just have the fun a
6 year old deserves.nannan
==================================================
```

In [14]:

```
# 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 [15]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out
for my students. I teach in a Title I school where most of the students receive free or reduced pr
ice lunch.  Despite their disabilities and limitations, my students love coming to school and come
eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to gr
oove and move as you were in a meeting? This is how my kids feel all the time. The want to be able
to move as they learn or so they say.Wobble chairs are the answer and I love then because they dev
elop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to l
earn through games, my kids do not want to sit and do worksheets. They want to learn to count by j
umping and playing. Physical engagement is the key to our success. The number toss and color and s
hape mats can make that happen. My students will forget they are doing work and just have the fun
a 6 year old deserves.nannan
==================================================
```

In [16]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays, cogniti
ve delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work th
eir hardest working past their limitations.     The materials we have are the ones I seek out for
my students. I teach in a Title I school where most of the students receive free or reduced price
lunch.  Despite their disabilities and limitations, my students love coming to school and come eag
er to learn and explore.Have you ever felt like you had ants in your pants and you needed to groov
e and move as you were in a meeting? This is how my kids feel all the time. The want to be able to
move as they learn or so they say.Wobble chairs are the answer and I love then because they develo
p their core, which enhances gross motor and in Turn fine motor skills.   They also want to learn t
hrough games, my kids do not want to sit and do worksheets. They want to learn to count by jumping
and playing. Physical engagement is the key to our success. The number toss and color and shape ma
ts can make that happen. My students will forget they are doing work and just have the fun a 6 yea
r old deserves.nannan
```

In [17]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitiv
e delays gross fine motor delays to autism They are eager beavers and always strive to work their
hardest working past their limitations The materials we have are the ones I seek out for my studen
ts I teach in a Title I school where most of the students receive free or reduced price lunch
Despite their disabilities and limitations my students love coming to school and come eager to lea
rn and explore Have you ever felt like you had ants in your pants and you needed to groove and mov
e as you were in a meeting This is how my kids feel all the time The want to be able to move as th
ey learn or so they say Wobble chairs are the answer and I love then because they develop their co
re which enhances gross motor and in Turn fine motor skills They also want to learn through games
my kids do not want to sit and do worksheets They want to learn to count by jumping and playing Ph
ysical engagement is the key to our success The number toss and color and shape mats can make that
happen My students will forget they are doing work and just have the fun a 6 year old deserves nan
nan
```

In [18]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
```

```
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [19]:

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

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000 [00:
55<00:00, 897.51it/s]
```

In [20]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[20]:

```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gros
s fine motor delays autism they eager beavers always strive work hardest working past limitations
the materials ones i seek students i teach title i school students receive free reduced price lunc
h despite disabilities limitations students love coming school come eager learn explore have ever
felt like ants pants needed groove move meeting this kids feel time the want able move learn say w
obble chairs answer i love develop core enhances gross motor turn fine motor skills they also want
learn games kids not want sit worksheets they want learn count jumping playing physical engagement
key success the number toss color shape mats make happen my students forget work fun 6 year old de
serves nannan'
```

1.6 Preprocessing of `project_title`

In [21]:

```
# Displaying first two datasets
X.head(2)
```

Out[21]:

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

In [22]:

```
# printing some random project titles.
print(X['project_title'].values[0])
print("="*50)
print(X['project_title'].values[150])
print("="*50)
print(X['project_title'].values[1000])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
```

In [23]:

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

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

    # 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]:

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

```
We GRIT If want meet tenacious respectful seven year olds growth mindsets need come classroom We g
ive hugs high fives compliments We Begin End Mind work hard everyday reach goals We not believe ma
king excuses times life need ask help As classroom teacher low income high poverty school district
2nd grade students face real life struggles classroom Even though visitor classroom would not know
daily struggle I ask How learn belly growling How I provide absolute best learning environment not
money buy research based materials Education not filling pail lighting fire William Butler Yeats W
```

money buy research based materials education not filling pail lighting fire william butler yeats w
e not asking fill pail things help provide resources light fire young minds Receiving books
written author teach students develop Writer Craft It inspire think different ways established aut
hors developed successful text appeal various audiences We never forget first love My mother read
Berenstain Bears series I five I fell love Berenstain family She took public library every week I
would hunt books written Stan Jan Berenstain Next curious monkey man yellow hat Curious George Tha
nk Margaret H A Rey creating series captured heart attention As teacher hope dream inspire student
s classroom find first love reading Help help discover writer craft go adventures minds develop
tenacious love reading sake reading nannan

In [25]:

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

we grit if want meet tenacious respectful seven year olds growth mindsets need come classroom we g
ive hugs high fives compliments we begin end mind work hard everyday reach goals we not believe ma
king excuses times life need ask help as classroom teacher low income high poverty school district
2nd grade students face real life struggles classroom even though visitor classroom would not know
daily struggle i ask how learn belly growling how i provide absolute best learning environment not
money buy research based materials education not filling pail lighting fire william butler yeats w
e not asking fill pail things help provide resources light fire young minds receiving books
written author teach students develop writer craft it inspire think different ways established aut
hors developed successful text appeal various audiences we never forget first love my mother read
berenstain bears series i five i fell love berenstain family she took public library every week i
would hunt books written stan jan berenstain next curious monkey man yellow hat curious george tha
nk margaret h a rey creating series captured heart attention as teacher hope dream inspire student
s classroom find first love reading help help discover writer craft go adventures minds develop
tenacious love reading sake reading nannan

In [26]:

```
sent = ' '.join(e for e in sent.split() if e not in stopwords)
print(sent)
```

grit want meet tenacious respectful seven year olds growth mindsets need come classroom give hugs
high fives compliments begin end mind work hard everyday reach goals not believe making excuses ti
mes life need ask help classroom teacher low income high poverty school district 2nd grade student
s face real life struggles classroom even though visitor classroom would not know daily struggle a
sk learn belly growling provide absolute best learning environment not money buy research based ma
terials education not filling pail lighting fire william butler yeats not asking fill pail things
help provide resources light fire young minds receiving books written author teach students
develop writer craft inspire think different ways established authors developed successful text ap
peal various audiences never forget first love mother read berenstain bears series five fell love
berenstain family took public library every week would hunt books written stan jan berenstain next
curious monkey man yellow hat curious george thank margaret h rey creating series captured heart a
ttention teacher hope dream inspire students classroom find first love reading help help discover
writer craft go adventures minds develop tenacious love reading sake reading nannan

In [27]:

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

100%|██████████| 50000/50000
[00:02<00:00, 20387.53it/s]

In [28]:

```
# after preprocesing
```

```
preprocessed_project_titles[20000]
```

Out[28]:

```
'need move input'
```

2.1 Preparing data for models

In [29]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

In [30]:

```
X_train=X_train.reset_index()
X_test=X_test.reset_index()
X_cv=X_cv.reset_index()
```

2.2 Preparing data for models

In [31]:

```
X.columns
```

Out[31]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', '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_approved',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'essay'],
      dtype='object')
```

we are going to consider

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

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

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

2.3 Vectorizing Categorical data

Respone coding

In [32]:

```
def response_coding_feature(alpha,feature,df):
    value_count=df[feature].value_counts()
    response_dict=dict()
```

```
    for i,denominator in value_count.items():
        vect=[];
        # k is considered in range(0,2) as only two classes are present
        for k in range(0,2):
            cls_count=df.loc[(df['project_is_approved']==k) & (df[feature]==i)]
            vect.append((cls_count.shape[0]+alpha*10)/(denominator+20*alpha))
        response_dict[i]=vect;

    response_feature=[]
    print(response_dict)
    for index,rows in df.iterrows():
        if rows[feature] in dict(value_count).keys():
            response_feature.append(response_dict[rows[feature]])
        else:
            response_feature.append([1/2,1/2])
    return response_feature;
```

In [33]:

```
from scipy.sparse import coo_matrix
alpha=1;
X_train_state_response_coding=np.array(response_coding_feature(alpha,"school_state",X_train))
X_cv_state_response_coding=np.array(response_coding_feature(alpha,"school_state",X_cv))
X_test_state_response_coding=np.array(response_coding_feature(alpha,"school_state",X_test))
```

```
{'CA': [0.1462882096069869, 0.8537117903930131], 'NY': [0.1571152607855763, 0.8428847392144238],
'TX': [0.19210526315789472, 0.8078947368421052], 'FL': [0.1703590527119939, 0.8296409472880061], '
NC': [0.15165441176470587, 0.8483455882352942], 'IL': [0.16252821670428894, 0.837471783295711], 'G
A': [0.1864801864801865, 0.8135198135198135], 'SC': [0.16646115906288533, 0.8335388409371147],
'MI': [0.17794117647058824, 0.8220588235294117], 'PA': [0.15477996965098634, 0.8452200303490136],
'MO': [0.14616755793226383, 0.8538324420677362], 'OH': [0.13059033989266547, 0.8694096601073346],
'IN': [0.14583333333333334, 0.8541666666666666], 'LA': [0.1932938856015779, 0.8067061143984221], '
MA': [0.1592741935483871, 0.8407258064516129], 'AZ': [0.18052738336713997, 0.8194726166328601], 'N
J': [0.1894093686354379, 0.810590631364562l], 'OK': [0.1629327902240326, 0.8370672097759674],
'WA': [0.11934156378600823, 0.8806584362139918], 'VA': [0.15813953488372093, 0.8418604651162791],
'WI': [0.1760204081632653, 0.8239795918367347], 'UT': [0.17819148936170212, 0.8218085106382979], '
AL': [0.15760869565217392, 0.8423913043478261], 'CT': [0.1392757660167131, 0.86072423398328691], 'TN
': [0.17847025495750707, 0.8215297450424929], 'NV': [0.159375, 0.840625], 'MD':
[0.1893687707641196, 0.8106312292358804], 'MS': [0.22, 0.78], 'KY': [0.15789473684210525,
0.8421052631578947], 'MN': [0.16541353383458646, 0.8345864661654135], 'OR': [0.20754716981132076,
0.7924528301886793], 'CO': [0.1891891891891892, 0.8108108108108109], 'AR': [0.19004524886877827, 0
.8099547511312217], 'IA': [0.20245398773006135, 0.7975460122699386], 'ID': [0.20253164556962025, 0
.7974683544303798], 'ME': [0.19014084507042253, 0.8098591549295775], 'NM': [0.16901408450704225, 0
.8309859154929577], 'KS': [0.19858156028368795, 0.8014184397163121], 'WV': [0.1746031746031746,
0.8253968253968254], 'DC': [0.22764227642276422, 0.7723577235772358], 'HI': [0.23008849557522124,
0.7699115044247787], 'DE': [0.22448979591836735, 0.7755102040816326], 'NH': [0.20930232558139536,
0.7906976744186046], 'NE': [0.2976190476190476, 0.7023809523809523], 'SD': [0.21052631578947367, 0
.7894736842105263], 'AK': [0.2894736842105263, 0.7105263157894737], 'RI': [0.25333333333333335,
0.7466666666666667], 'MT': [0.36507936507936506, 0.6349206349206349], 'ND': [0.3111111111111111, 0
.6888888888888889], 'WY': [0.3170731707317073, 0.6829268292682927], 'VT': [0.3611111111111111,
0.6388888888888888]}
{'CA': [0.15227127319257838, 0.8477287268074216], 'NY': [0.14035087719298245, 0.8596491228070176],
'TX': [0.19764397905759162, 0.8023560209424084], 'FL': [0.1935483870967742, 0.8064516129032258], '
NC': [0.15643564356435644, 0.8435643564356435], 'IL': [0.16421052631578947, 0.8357894736842105], '
SC': [0.15973741794310722, 0.8402625820568927], 'GA': [0.16176470588235295, 0.8382352941176471], '
MI': [0.17955801104972377, 0.8204419889502762], 'PA': [0.1676300578034682, 0.8323699421965318], 'O
H': [0.13559322033898305, 0.864406779661017], 'MO': [0.1773049645390071, 0.8226950354609929],
'LA': [0.16605166051660517, 0.8339483394833949], 'OK': [0.20149253731343283, 0.7985074626865671],
'MA': [0.17228464419475656, 0.8277153558052435], 'IN': [0.2277992277992278, 0.7722007722007722], '
WA': [0.14457831325301204, 0.8554216867469879], 'NJ': [0.22357723577235772, 0.7764227642276422], '
VA': [0.1889400921658986, 0.8110599078341014], 'AZ': [0.2037037037037037, 0.7962962962962963],
'TN': [0.18536585365853658, 0.8146341463414634], 'CT': [0.17733990147783252, 0.8226600985221675],
'AL': [0.19801980198019803, 0.8019801980198019802], 'WI': [0.1881720430107527, 0.8118279569892473], 'U
T': [0.19318181818181818, 0.8068181818181818], 'MD': [0.20121951219512196, 0.7987804878048781], 'N
V': [0.189873417721519, 0.810126582278481], 'KY': [0.17763157894736842, 0.8223684210526315], 'OR':
[0.23178807947019867, 0.7682119205298014], 'MS': [0.1456953642384106, 0.8543046357615894], 'CO':
[0.22758620689655173, 0.7724137931034483], 'MN': [0.2189781021897810 3, 0.781021897810219], 'AR': [
0.20535714285714285, 0.7946428571428571], 'IA': [0.2608695652173913, 0.7391304347826086], 'KS':
[0.1744186046511628, 0.8255813953488372], 'HI': [0.2, 0.8], 'DC': [0.30120481927710846,
0.6987951807228916], 'ID': [0.2926829268292683, 0.7073170731707317], 'NM': [0.26153846153846155, 0
.7384615384615385], 'ME': [0.3548387096774194, 0.6451612903225806], 'WV': [0.26666666666666666,
0.7333333333333333], 'DE': [0.2, 0.8], 'RI': [0.2631578947368421, 0.7368421052631579], 'NH':
[0.25925925925925924, 0.7407407407407407], 'AK': [0.27450980392156865, 0.7254901960784313], 'SD':
[0.3, 0.7], 'NE': [0.3125, 0.6875], 'MT': [0.3, 0.7], 'WY': [0.3225806451612903,
```

```
0.6774193548387096], 'ND': [0.3225806451612903, 0.6774193548387096], 'VT': [0.42857142857142855, 0
.5714285714285714]}
{'CA': [0.14427645788336932, 0.8557235421166307], 'NY': [0.14882032667876588, 0.8511796733212341],
'TX': [0.21715328467153286, 0.7828467153284672], 'FL': [0.20412371134020618, 0.7958762886597938],
'NC': [0.14869888475836432, 0.8513011152416357], 'IL': [0.17417417417417416, 0.8258258258258259],
'GA': [0.15755627009646303, 0.842443729903537], 'SC': [0.1382636655948553, 0.8617363344051447], 'M
I': [0.16666666666666666, 0.8333333333333334], 'PA': [0.15822784810126583, 0.8417721518987342], 'I
N': [0.17792792792792791, 0.8220720720720721], 'WA': [0.1378504672897196, 0.8621495327102804],
'OH': [0.15803108808290156, 0.8419689119170984], 'MO': [0.15926892950391644, 0.8407310704960835],
'LA': [0.19414893617021275, 0.8058510638297872], 'OK': [0.16533333333333333, 0.8346666666666667],
'MA': [0.1742627345844504, 0.8257372654155496], 'AZ': [0.16231884057971013, 0.8376811594202899], '
VA': [0.1702127659574468, 0.8297872340425532], 'NJ': [0.18292682926829268, 0.8170731707317073], 'W
I': [0.19365079365079366, 0.8063492063492064], 'UT': [0.19666666666666666, 0.8033333333333333], 'A
L': [0.16785714285714284, 0.8321428571428572], 'TN': [0.17753623188405798, 0.822463768115942], 'CT
': [0.12867647058823528, 0.8713235294117647], 'MD': [0.1596958174904943, 0.8403041825095057],
'NV': [0.16194331983805668, 0.8380566801619433], 'KY': [0.16033755274261605, 0.8396624472573839],
'OR': [0.19457013574660634, 0.8054298642533937], 'MN': [0.17370892018779344, 0.8262910798122066],
'MS': [0.18357487922705315, 0.8164251207729468], 'CO': [0.1958762886597938, 0.8041237113402062], '
AR': [0.23121387283236994, 0.7687861271676301], 'ID': [0.27049180327868855, 0.7295081967213115], '
KS': [0.1864406779661017, 0.8135593220338984], 'IA': [0.18018018018018017, 0.8198198198198198], 'H
I': [0.21782178217821782, 0.7821782178217822], 'DC': [0.297029702970297, 0.7029702970297029],
'WV': [0.2391304347826087, 0.7608695652173914], 'NM': [0.2247191011235955, 0.7752808988764045],
'AK': [0.2441860465116279, 0.7558139534883721], 'ME': [0.23076923076923078, 0.7692307692307693], '
SD': [0.2631578947368421, 0.7368421052631579], 'NE': [0.19444444444444445, 0.8055555555555556], 'M
T': [0.31746031746031744, 0.6825396825396826], 'NH': [0.2459016393442623, 0.7540983606557377],
'DE': [0.2631578947368421, 0.7368421052631579], 'RI': [0.2777777777777778, 0.7222222222222222],
'ND': [0.2553191489361702, 0.7446808510638298], 'WY': [0.41025641025641024, 0.5897435897435898], '
VT': [0.42857142857142855, 0.5714285714285714]}
```

In [34]:

```
alpha=1;
X_train_teacher_prefix_response_coding=np.array(response_coding_feature(alpha,"teacher_prefix",X_t
rain))
X_cv_teacher_prefix_response_coding=np.array(response_coding_feature(alpha,"teacher_prefix",X_cv))
X_test_teacher_prefix_response_coding=np.array(response_coding_feature(alpha,"teacher_prefix",X_tes
t))
```

```
{'Mrs.': [0.14917173766058148, 0.8508282623394186], 'Ms.': [0.1580262336039975,
0.8419737663960025], 'Mr.': [0.16099773242630386, 0.8390022675736961], 'Teacher':
[0.2443064182194617, 0.7556935817805382]}
{'Mrs.': [0.1497651765524439, 0.8502348234475561], 'Ms.': [0.16021825396825398,
0.839781746031746], 'Mr.': [0.17582417582417584, 0.8241758241758241], 'Teacher':
[0.17624521072796934, 0.8237547892720306], 'Dr.': [0.47619047619047616, 0.5238095238095238]}
{'Mrs.': [0.15140967629655414, 0.8485903237034459], 'Ms.': [0.15623426749454605,
0.843765732505454], 'Mr.': [0.1658446362515413, 0.8341553637484587], 'Teacher':
[0.21220159151193635, 0.7877984084880637], 'Dr.': [0.5238095238095238, 0.47619047619047616]}
```

In [35]:

```
alpha=1;
X_train_project_grade_category_response_coding=np.array(response_coding_feature(alpha,"project_grac
e_category",X_train))
X_cv_project_grade_category_response_coding=np.array(response_coding_feature(alpha,"project_grade_c
ategory",X_cv))
X_test_project_grade_category_response_coding=np.array(response_coding_feature(alpha,"project_grade
_category",X_test))
```

```
{'Grades_PreKTo2': [0.15170546832701678, 0.8482945316729832], 'Grades_3To5': [0.14865577227200844,
0.8513442772779916], 'Grades_6To8': [0.1693106432073839, 0.8306893567926161], 'Grades_9To12':
[0.17315436241610738, 0.8268456375838926]}
{'Grades_PreKTo2': [0.15225605690153368, 0.8477439430984663], 'Grades_3To5': [0.1515310128238681,
0.8484689871176132], 'Grades_6To8': [0.16292798110979928, 0.8370720188902007], 'Grades_9To12':
[0.18287243532560213, 0.8171275646743978]}
{'Grades_PreKTo2': [0.16184884071062933, 0.8381511592893707], 'Grades_3To5': [0.14682327816337426,
0.8531767218366257], 'Grades_6To8': [0.15628539071347677, 0.8437146092865232], 'Grades_9To12': [0.
162874251497006, 0.837125748502994]}
```

In [36]:

```
alpha=1;
X_train_clean_categories_response_coding=np.array(response_coding_feature(alpha,"clean_categories"
,X_train))
```

```
X_cv_clean_categories_response_coding=np.array(response_coding_feature(alpha,"clean_categories",X_
cv))
X_test_clean_categories_response_coding=np.array(response_coding_feature(alpha,"clean_categories",
X_test))
```


{'Literacy_Language': [0.13954434499593166, 0.8604556550040684], 'Math_Science': [0.17952478732766208, 0.8204752126723379], 'Literacy_Language Math_Science': [0.13457760314341846, 0.8654223968565815], 'Health_Sports': [0.157502329916123, 0.842497670083877], 'Music_Arts': [0.14947965941343425, 0.85052034058656581, 'SpecialNeeds': [0.1948488241881299, 0.8051511758118701], 'Literacy_Language SpecialNeeds': [0.15042117930204574, 0.8495788206979543], 'AppliedLearning': [0.1889763779527559, 0.8110236220472441], 'Math_Science Literacy_Language': [0.15352697095435686, 0.8464730290456431], 'AppliedLearning Literacy_Language': [0.16632016632016633, 0.8336798336798337], 'Math_Science SpecialNeeds': [0.17703349282296652, 0.8229665071770335], 'Literacy_Language Music_Arts': [0.19170984455958548, 0.8082901554404145], 'History_Civics': [0.20680628272251309, 0.7931937172774869], 'Math_Science Music_Arts': [0.18498659517426275, 0.8150134048257373], 'AppliedLearning SpecialNeeds': [0.22658610271903323, 0.7734138972809668], 'History_Civics Literacy_Language': [0.13271604938271606, 0.8672839506172839], 'Warmth Care_Hunger': [0.10135135135135136, 0.8986486486486487], 'Health_Sports SpecialNeeds': [0.17006802721088435, 0.8299319727891157], 'Math_Science AppliedLearning': [0.18181818181818182, 0.8181818181818182], 'AppliedLearning Math_Science': [0.2026431718061674, 0.7973568281938326], 'Health_Sports Literacy_Language': [0.22959183673469388, 0.7704081632653061], 'Literacy_Language History_Civics': [0.17708333333333334, 0.8229166666666666], 'AppliedLearning Music_Arts': [0.2459016393442623, 0.7540983606557377], 'Math_Science History_Civics': [0.16560509554140126, 0.8343949044585988], 'Literacy_Language AppliedLearning': [0.19444444444444445, 0.8055555555555556], 'AppliedLearning Health_Sports': [0.21481481481481482, 0.7851851851851852], 'Math_Science Health_Sports': [0.2767857142857143, 0.7232142857142857], 'History_Civics Math_Science': [0.19230769230769232, 0.8076923076923077], 'SpecialNeeds Music_Arts': [0.3411764705882353, 0.6588235294117647], 'History_Civics Music_Arts': [0.2823529411764706, 0.7176470588235294], 'Health_Sports Math_Science': [0.3384615384615385, 0.6615384615384615], 'History_Civics SpecialNeeds': [0.24615384615384617, 0.7538461538461538], 'Health_Sports AppliedLearning': [0.3559322033898305, 0.6440677966101694], 'AppliedLearning History_Civics': [0.2711864406779661, 0.7288135593220338], 'Health_Sports Music_Arts': [0.3333333333333333, 0.6666666666666666], 'Music_Arts SpecialNeeds': [0.2653061224489796, 0.7346938775510204], 'Literacy_Language Health_Sports': [0.32432432432432434, 0.6756756756756757], 'History_Civics AppliedLearning': [0.35294117647058826, 0.6470588235294118], 'Health_Sports History_Civics': [0.36666666666666664, 0.6333333333333333], 'Music_Arts Health_Sports': [0.4074074074074074, 0.5925925925925926], 'Health_Sports Warmth Care_Hunger': [0.4230769230769231, 0.5769230769230769], 'Music_Arts History_Civics': [0.52, 0.48], 'Music_Arts AppliedLearning': [0.43478260869565216, 0.5652173913043478], 'History_Civics Health_Sports': [0.43478260869565216, 0.5652173913043478], 'AppliedLearning Warmth Care_Hunger': [0.43478260869565216, 0.5652173913043478], 'SpecialNeeds Health_Sports': [0.5, 0.5], 'Math_Science Warmth Care_Hunger': [0.5, 0.5], 'Literacy_Language Warmth Care_Hunger': [0.45454545454545453, 0.5454545454545454], 'SpecialNeeds Warmth Care_Hunger': [0.47619047619047616, 0.5238095238095238], 'Music_Arts Warmth Care_Hunger': [0.5238095238095238, 0.47619047619047616]}
{'Literacy_Language': [0.1474120082815735, 0.8525879917184265], 'Math_Science': [0.186558516801854, 0.813441483198146], 'Literacy_Language Math_Science': [0.13232389730085584, 0.8676761026991442], 'Health_Sports': [0.1517509727626459, 0.8482490272373541], 'Music_Arts': [0.161231884057971, 0.8387681159420289], 'SpecialNeeds': [0.1772428884026258, 0.8227571115973742], 'Literacy_Language SpecialNeeds': [0.16397228637413394, 0.836027713625866], 'AppliedLearning': [0.22305764411027568, 0.7769423558897243], 'AppliedLearning Literacy_Language': [0.18992248062015504, 0.810077519379845], 'Math_Science Literacy_Language': [0.15873015873015872, 0.8412698412698413], 'History_Civics': [0.1784037558685446, 0.8215962441314554], 'Math_Science SpecialNeeds': [0.23115577889447236, 0.7688442211055276], 'Math_Science Music_Arts': [0.20418848167539266, 0.7958115183246073], 'Literacy_Language Music_Arts': [0.20930232558139536, 0.7906976744186046], 'Health_Sports SpecialNeeds': [0.19047619047619047, 0.8095238095238095], 'Warmth Care_Hunger': [0.1337579617834395, 0.8662420382165605], 'History_Civics Literacy_Language': [0.13548387096774195, 0.864516129032258], 'AppliedLearning SpecialNeeds': [0.2080536912751678, 0.7919463087248322], 'Math_Science AppliedLearning': [0.1956521739130435, 0.8043478260869565], 'AppliedLearning Math_Science': [0.2777777777777778, 0.7222222222222222], 'Health_Sports Literacy_Language': [0.23529411764705882, 0.7647058823529411], 'Literacy_Language History_Civics': [0.15, 0.85], 'AppliedLearning Music_Arts': [0.22340425531914893, 0.776595744680851], 'AppliedLearning Health_Sports': [0.23333333333333334, 0.7666666666666667], 'Literacy_Language AppliedLearning': [0.2597402597402597, 0.7402597402597403], 'Math_Science History_Civics': [0.27631578947368424, 0.7236842105263158], 'Math_Science Health_Sports': [0.2698412698412698, 0.7301587301587301], 'Health_Sports Math_Science': [0.3389830508474576, 0.6610169491525424], 'History_Civics Math_Science': [0.2982456140350877, 0.7017543859649122], 'History_Civics SpecialNeeds': [0.39622641509433965, 0.6037735849056604], 'SpecialNeeds Music_Arts': [0.29411764705882354, 0.7058823529411765], 'History_Civics Music_Arts': [0.26, 0.74], 'Health_Sports AppliedLearning': [0.2916666666666667, 0.7083333333333334], 'Music_Arts SpecialNeeds': [0.32432432432432434, 0.6756756756756757], 'AppliedLearning History_Civics': [0.2972972972972973, 0.7027027027027027], 'Health_Sports Music_Arts': [0.41935483870967744, 0.5806451612903226], 'Health_Sports History_Civics': [0.4074074074074074, 0.5925925925925926], 'Literacy_Language Health_Sports': [0.4444444444444444, 0.5555555555555556], 'History_Civics AppliedLearning': [0.5, 0.5], 'SpecialNeeds Health_Sports': [0.44, 0.56], 'SpecialNeeds Warmth Care_Hunger': [0.44, 0.56], 'AppliedLearning Warmth Care_Hunger': [0.4782608695652174, 0.5217391304347826], 'Music_Arts History_Civics': [0.43478260869565216, 0.5652173913043478], 'Math_Science Warmth Care_Hunger': [0.45454545454545453, 0.5454545454545454], 'Health_Sports Warmth Car

e_Hunger': [0.45454545454545453, 0.5454545454545454], 'History_Civics Health_Sports':
[0.45454545454545453, 0.5454545454545454], 'Music_Arts Health_Sports': [0.5238095238095238,
0.47619047619047616]}
{'Literacy_Language': [0.1288293216630197, 0.8711706783369803], 'Math_Science':
[0.18473282442748093, 0.815267175572519], 'Literacy_Language Math_Science': [0.14507299270072993,
0.8549270072992701], 'Health_Sports': [0.16393442622950818, 0.8360655737704918], 'Music_Arts': [0.
14833127317676142, 0.8516687268232386], 'SpecialNeeds': [0.21348314606741572, 0.7865168539325843],
'AppliedLearning': [0.20064724919093851, 0.7993527508090615], 'Literacy_Language SpecialNeeds': [0
.16229508196721312, 0.8377049180327869], 'Math_Science Literacy_Language': [0.14168937329700274, 0
.8583106267029973], 'AppliedLearning Literacy_Language': [0.168141592920354, 0.831858407079646], '
Math_Science SpecialNeeds': [0.21656050955414013, 0.7834394904458599], 'History_Civics':
[0.20723684210526316, 0.7927631578947368], 'Literacy_Language Music_Arts': [0.1891891891891892,
0.8108108108108109], 'AppliedLearning SpecialNeeds': [0.21428571428571427, 0.7857142857142857], 'M
ath_Science Music_Arts': [0.20717131474103587, 0.7928286852589641], 'History_Civics
Literacy_Language': [0.09482758620689655, 0.9051724137931034], 'Health_Sports SpecialNeeds':
[0.1471861471861472, 0.8528138528138528], 'Warmth Care_Hunger': [0.11267605633802817,
0.8873239436619719], 'Math_Science AppliedLearning': [0.208955223880597, 0.7910447761194029],
'AppliedLearning Math_Science': [0.21739130434782608, 0.782608695652174], 'AppliedLearning
Music_Arts': [0.23776223776223776, 0.7622377622377622], 'Health_Sports Literacy_Language':
[0.20610687022900764, 0.7938931297709924], 'Literacy_Language History_Civics':
[0.19083969465648856, 0.8091603053435115], 'Literacy_Language AppliedLearning':
[0.226890756302521, 0.773109243697479], 'Math_Science History_Civics': [0.24770642201834864,
0.7522935779816514], 'AppliedLearning Health_Sports': [0.23232323232323232, 0.7676767676767676], '
Math_Science Health_Sports': [0.3333333333333333, 0.6666666666666666], 'History_Civics
Math_Science': [0.2571428571428571, 0.7428571428571429], 'SpecialNeeds Music_Arts': [0.1875,
0.8125], 'History_Civics Music_Arts': [0.25, 0.75], 'Health_Sports Math_Science':
[0.2962962962962963, 0.7037037037037037], 'Health_Sports AppliedLearning': [0.2692307692307692, 0.
7307692307692307], 'History_Civics SpecialNeeds': [0.35555555555555557, 0.6444444444444445],
'Health_Sports Music_Arts': [0.38636363636363635, 0.6136363636363636], 'AppliedLearning
History_Civics': [0.42857142857142855, 0.5714285714285714], 'Music_Arts SpecialNeeds':
[0.2682926829268293, 0.7317073170731707], 'Literacy_Language Health_Sports': [0.41379310344827586,
0.5862068965517241], 'Health_Sports History_Civics': [0.35714285714285715, 0.6428571428571429], 'S
pecialNeeds Health_Sports': [0.4444444444444444, 0.5555555555555556], 'History_Civics
AppliedLearning': [0.44, 0.56], 'Health_Sports Warmth Care_Hunger': [0.4166666666666667,
0.5833333333333334], 'Math_Science Warmth Care_Hunger': [0.4782608695652174, 0.5217391304347826],
'History_Civics Health_Sports': [0.43478260869565216, 0.5652173913043478], 'Music_Arts
Health_Sports': [0.45454545454545453, 0.5454545454545454], 'AppliedLearning Warmth Care_Hunger': [
0.5, 0.5], 'Music_Arts AppliedLearning': [0.5238095238095238, 0.47619047619047616],
'Literacy_Language Warmth Care_Hunger': [0.47619047619047616, 0.5238095238095238], 'Music_Arts His
tory_Civics': [0.5238095238095238, 0.47619047619047616]}

In [37]:

```
alpha=1;
X_train_clean_subcategories_response_coding=np.array(response_coding_feature(alpha,"clean_subcategc
ries",X_train))
X_cv_clean_subcategories_response_coding=np.array(response_coding_feature(alpha,"clean_subcategorie
s",X_cv))
X_test_clean_subcategories_response_coding=np.array(response_coding_feature(alpha,"clean_subcategor
ies",X_test))
```

{'Literacy': [0.12406576980568013, 0.8759342301943199], 'Literacy Mathematics':
[0.14189189189189189, 0.8581081081081081], 'Literature_Writing Mathematics': [0.12489728841413311,
0.8751027115858668], 'Literacy Literature_Writing': [0.13529921942758022, 0.8647007805724197], 'Ma
thematics': [0.16925892040256177, 0.8307410795974383], 'Literature_Writing': [0.1792152704135737,
0.8207847295864263], 'SpecialNeeds': [0.1948488241881299, 0.8051511758118701], 'Health_Wellness':
[0.14766839378238342, 0.8523316062176166], 'AppliedSciences Mathematics': [0.1902834008097166,
0.8097165991902834], 'Literacy SpecialNeeds': [0.13142857142857142, 0.8685714285714285],
'Gym_Fitness Health_Wellness': [0.12959381044487428, 0.8704061895551257], 'AppliedSciences':
[0.1894093686354379, 0.8105906313645621], 'ESL Literacy': [0.16907216494845362,
0.8309278350515464], 'VisualArts': [0.17516629711751663, 0.8248337028824834], 'Music': [0.15, 0.85
], 'Warmth Care_Hunger': [0.10135135135135136, 0.8986486486486487], 'Literature_Writing
SpecialNeeds': [0.2108843537414966, 0.7891156462585034], 'Mathematics SpecialNeeds':
[0.19852941176470587, 0.8014705882352942], 'EnvironmentalScience': [0.2159090909090909,
0.7840909090909091], 'Gym_Fitness': [0.18604651162790697, 0.813953488372093], 'Health_Wellness Spe
cialNeeds': [0.1732283464566929, 0.8267716535433071], 'TeamSports': [0.1759656652360515,
0.8240343347639485], 'Music PerformingArts': [0.16744186046511628, 0.8325581395348837],
'EarlyDevelopment': [0.22488038277511962, 0.7751196172248804], 'EnvironmentalScience
Health_LifeScience': [0.23414634146341465, 0.7658536585365854], 'Other': [0.1715686274509804, 0.82
84313725490197], 'AppliedSciences EnvironmentalScience': [0.22885572139303484,
0.7711442786069652], 'EarlyDevelopment SpecialNeeds': [0.211340206185567, 0.788659793814433], 'ESL
Literature_Writing': [0.1693121693121693, 0.8306878306878307], 'Health_Wellness
NutritionEducation': [0.24456521739130435, 0.7554347826086957], 'Health_LifeScience':
[0.2057142857142857, 0.7942857142857143], 'AppliedSciences VisualArts': [0.2, 0.8],
'EnvironmentalScience Mathematics': [0.18235294117647058, 0.8176470588235294], 'Literature_Writing

VisualArts': [0.23668639053254437, 0.7633136094674556], 'EarlyDevelopment Literacy':
[0.1952662721893491, 0.8047337278106509], 'AppliedSciences Literacy': [0.22818791946308725,
0.7718120805369127], 'History_Geography Literature_Writing': [0.2054794520547945,
0.7945205479452054], 'Gym_Fitness TeamSports': [0.2638888888888889, 0.7361111111111112],
'AppliedSciences Health_LifeScience': [0.23703703703703705, 0.762962962962963], 'History_Geography
Literacy': [0.13076923076923078, 0.8692307692307693], 'Health_Wellness Literacy':
[0.25196850393700787, 0.7480314960629921], 'Mathematics VisualArts': [0.224, 0.776], 'Literacy
VisualArts': [0.19834710743801653, 0.8016528925619835], 'History_Geography': [0.2711864406779661,
0.7288135593220338], 'AppliedSciences College_CareerPrep': [0.23478260869565218,
0.7652173913043478], 'Health_LifeScience Mathematics': [0.2894736842105263, 0.7105263157894737], '
AppliedSciences Literature_Writing': [0.18518518518518517, 0.8148148148148148], 'Literacy
SocialSciences': [0.21904761904761905, 0.780952380952381], 'ESL': [0.23076923076923078,
0.7692307692307693], 'PerformingArts': [0.17475728155339806, 0.8252427184466019],
'EnvironmentalScience Literacy': [0.18627450980392157, 0.8137254901960784], 'AppliedSciences
SpecialNeeds': [0.2268041237113402, 0.7731958762886598], 'Literature_Writing SocialSciences':
[0.1958762886597938, 0.8041237113402062], 'CharacterEducation Literacy': [0.21839080459770116, 0.7
8160919540229889], 'CharacterEducation': [0.30232558139534888, 0.6976744186046512],
'ForeignLanguages': [0.24705882352941178, 0.7529411764705882], 'SpecialNeeds VisualArts':
[0.3411764705882353, 0.6588235294117647], 'Health_Wellness TeamSports': [0.2857142857142857,
0.7142857142857143], 'EarlyDevelopment Literature_Writing': [0.27710843373493976,
0.7228915662650602], 'College_CareerPrep': [0.26506024096385544, 0.7349397590361446],
'NutritionEducation': [0.34146341463414637, 0.6585365853658537], 'Health_Wellness
Literature_Writing': [0.24691358024691357, 0.7530864197530864], 'College_CareerPrep
Literature_Writing': [0.22784810126582278, 0.7721518987341772], 'EarlyDevelopment Mathematics': [0
.2948717948717949, 0.7051282051282052], 'ESL Mathematics': [0.2564102564102564,
0.7435897435897436], 'History_Geography SocialSciences': [0.28205128205128205, 0.717948717948718],
'Other SpecialNeeds': [0.3333333333333333, 0.6666666666666666], 'EarlyDevelopment
Health_Wellness': [0.23076923076923078, 0.7692307692307693], 'Health_LifeScience Literacy':
[0.2236842105263158, 0.7763157894736842], 'College_CareerPrep Literacy': [0.22666666666666666, 0.7
7333333333333333], 'College_CareerPrep Mathematics': [0.24324324324324326, 0.7567567567567568], 'Ci
vics_Government History_Geography': [0.28378378378378377, 0.7162162162162162],
'EnvironmentalScience Literature_Writing': [0.2361111111111111, 0.7638888888888888],
'ForeignLanguages Literacy': [0.25396825396825395, 0.746031746031746], 'CharacterEducation
SpecialNeeds': [0.26666666666666666, 0.7333333333333333], 'EnvironmentalScience VisualArts':
[0.3220338983050847, 0.6779661016949152], 'FinancialLiteracy': [0.288135593220339,
0.711864406779661], 'History_Geography VisualArts': [0.3389830508474576, 0.6610169491525424],
'Literacy Other': [0.2413793103448276, 0.7586206896551724], 'SocialSciences': [0.2982456140350877,
0.7017543859649122], 'EnvironmentalScience SpecialNeeds': [0.21052631578947367,
0.7894736842105263], 'AppliedSciences EarlyDevelopment': [0.30357142857142855,
0.6964285714285714], 'Health_LifeScience Health_Wellness': [0.30357142857142855,
0.6964285714285714], 'Health_LifeScience Literature_Writing': [0.21428571428571427,
0.7857142857142857], 'Civics_Government Literacy': [0.21428571428571427, 0.7857142857142857],
'Health_Wellness Mathematics': [0.375, 0.625], 'EarlyDevelopment VisualArts':
[0.35714285714285715, 0.6428571428571429], 'FinancialLiteracy Mathematics': [0.2909090909090909,
0.7090909090909091], 'EnvironmentalScience History_Geography': [0.2, 0.8], 'AppliedSciences
Extracurricular': [0.2222222222222222, 0.7777777777777778], 'Health_Wellness Other':
[0.37037037037037035, 0.6296296296296297], 'Gym_Fitness SpecialNeeds': [0.25925925925925924,
0.7407407407407407], 'Literacy ParentInvolvement': [0.3018867924528302, 0.6981132075471698],
'CharacterEducation Literature_Writing': [0.2692307692307692, 0.7307692307692307], 'Literacy
PerformingArts': [0.36538461538461536, 0.6346153846153846], 'Health_LifeScience SpecialNeeds': [0.
3076923076923077, 0.6923076923076923], 'CharacterEducation EarlyDevelopment':
[0.36538461538461536, 0.6346153846153846], 'ESL SpecialNeeds': [0.27450980392156865,
0.7254901960784313], 'EarlyDevelopment Other': [0.21568627450980393, 0.7843137254901961],
'History_Geography Mathematics': [0.2549019607843137, 0.7450980392156863], 'Literacy Music':
[0.26, 0.74], 'College_CareerPrep VisualArts': [0.30612244897959184, 0.6938775510204082],
'CharacterEducation Other': [0.3333333333333333, 0.6666666666666666], 'CharacterEducation
Mathematics': [0.2978723404255319, 0.7021276595744681], 'Music SpecialNeeds': [0.2608695652173913,
0.7391304347826086], 'History_Geography SpecialNeeds': [0.3111111111111111, 0.6888888888888889], '
College_CareerPrep SpecialNeeds': [0.4090909090909091, 0.5909090909090909], 'AppliedSciences
Other': [0.29545454545454547, 0.7045454545454546], 'Literature_Writing PerformingArts':
[0.3023255813953488, 0.6976744186046512], 'Literature_Writing Other': [0.2926829268292683,
0.7073170731707317], 'CharacterEducation College_CareerPrep': [0.36585365853658536,
0.6341463414634146], 'CharacterEducation Health_Wellness': [0.3170731707317073,
0.6829268292682927], 'Mathematics SocialSciences': [0.3, 0.7], 'Civics_Government
Literature_Writing': [0.3, 0.7], 'Extracurricular VisualArts': [0.3333333333333333,
0.6666666666666666], 'Economics FinancialLiteracy': [0.358974358974359, 0.6410256410256411],
'AppliedSciences Music': [0.3333333333333333, 0.6666666666666666], 'CharacterEducation
VisualArts': [0.34210526315789475, 0.6578947368421053], 'Civics_Government SocialSciences':
[0.3157894736842105, 0.6842105263157895], 'AppliedSciences History_Geography':
[0.34210526315789475, 0.6578947368421053], 'AppliedSciences Health_Wellness':
[0.34210526315789475, 0.6578947368421053], 'Health_LifeScience VisualArts': [0.35135135135135137,
0.6486486486486487], 'PerformingArts VisualArts': [0.3783783783783784, 0.6216216216216216],
'Mathematics Other': [0.32432432432432434, 0.6756756756756757], 'CharacterEducation
CommunityService': [0.32432432432432434, 0.6756756756756757], 'Extracurricular':
[0.3055555555555556, 0.6944444444444444], 'Health_LifeScience SocialSciences':
[0.3611111111111111, 0.6388888888888888], 'ESL EarlyDevelopment': [0.3333333333333333,
0.6666666666666666], 'Other VisualArts': [0.3888888888888889, 0.6111111111111112],

'EnvironmentalScience SocialSciences': [0.3611111111111111, 0.6388888888888888], 'Mathematics ParentInvolvement': [0.37142857142857144, 0.6285714285714286], 'AppliedSciences ESL': [0.37142857142857144, 0.6285714285714286], 'College_CareerPrep Other': [0.4, 0.6], 'ForeignLanguages Literature_Writing': [0.42857142857142855, 0.5714285714285714], 'CharacterEducation Extracurricular': [0.47058823529411764, 0.5294117647058824], 'AppliedSciences CharacterEducation': [0.35294117647058826, 0.6470588235294118], 'Mathematics Music': [0.3235294117647059, 0.6764705882352942], 'Extracurricular Mathematics': [0.35294117647058826, 0.6470588235294118], 'AppliedSciences ParentInvolvement': [0.35294117647058826, 0.6470588235294118], 'AppliedSciences SocialSciences': [0.3235294117647059, 0.6764705882352942], 'FinancialLiteracy SpecialNeeds': [0.36363636363636365, 0.6363636363636364], 'Literature_Writing Music': [0.30303030303030304, 0.696969696969697], 'Health_LifeScience History_Geography': [0.36363636363636365, 0.6363636363636364], 'EarlyDevelopment EnvironmentalScience': [0.3939393939393939, 0.6060606060606061], 'EnvironmentalScience Health_Wellness': [0.34375, 0.65625], 'ESL ForeignLanguages': [0.375, 0.625], 'Civics_Government': [0.34375, 0.65625], 'Gym_Fitness NutritionEducation': [0.40625, 0.59375], 'Music VisualArts': [0.41935483870967744, 0.5806451612903226], 'Health_LifeScience NutritionEducation': [0.45161290322580644, 0.5483870967741935], 'CommunityService': [0.36666666666666664, 0.6333333333333333], 'Literature_Writing ParentInvolvement': [0.4, 0.6], 'College_CareerPrep EnvironmentalScience': [0.4, 0.6], 'ESL VisualArts': [0.4482758620689655, 0.5517241379310345], 'College_CareerPrep SocialSciences': [0.41379310344827586, 0.5862068965517241], 'Economics': [0.3793103448275862, 0.6206896551724138], 'SocialSciences VisualArts': [0.3793103448275862, 0.6206896551724138], 'ESL Health_LifeScience': [0.5357142857142857, 0.4642857142857143], 'College_CareerPrep Extracurricular': [0.35714285714285715, 0.6428571428571429], 'Health_Wellness VisualArts': [0.42857142857142855, 0.5714285714285714], 'Economics Mathematics': [0.35714285714285715, 0.6428571428571429], 'College_CareerPrep ParentInvolvement': [0.42857142857142855, 0.5714285714285714], 'Extracurricular Literacy': [0.4642857142857143, 0.5357142857142857], 'Gym_Fitness Mathematics': [0.39285714285714285, 0.6071428571428571], 'ForeignLanguages Mathematics': [0.35714285714285715, 0.6428571428571429], 'EarlyDevelopment Gym_Fitness': [0.4074074074074074, 0.5925925925925926], 'EarlyDevelopment Health_LifeScience': [0.48148148148148145, 0.5185185185185185], 'ParentInvolvement VisualArts': [0.4074074074074074, 0.5925925925925926], 'CommunityService Literature_Writing': [0.4074074074074074, 0.5925925925925926], 'Economics History_Geography': [0.38461538461538464, 0.6153846153846154], 'College_CareerPrep PerformingArts': [0.4230769230769231, 0.5769230769230769], 'Health_Wellness Music': [0.4230769230769231, 0.5769230769230769], 'CommunityService SpecialNeeds': [0.46153846153846156, 0.5384615384615384], 'SocialSciences SpecialNeeds': [0.38461538461538464, 0.6153846153846154], 'Civics_Government CommunityService': [0.4230769230769231, 0.5769230769230769], 'College_CareerPrep FinancialLiteracy': [0.46153846153846156, 0.5384615384615384], 'ESL Health_Wellness': [0.38461538461538464, 0.6153846153846154], 'ESL History_Geography': [0.4230769230769231, 0.5769230769230769], 'History_Geography Music': [0.4230769230769231, 0.5769230769230769], 'Extracurricular Music': [0.4230769230769231, 0.5769230769230769], 'Civics_Government Economics': [0.4230769230769231, 0.5769230769230769], 'NutritionEducation SpecialNeeds': [0.46153846153846156, 0.5384615384615384], 'Health_Wellness Warmth Care_Hunger': [0.4230769230769231, 0.5769230769230769], 'ESL EnvironmentalScience': [0.46153846153846156, 0.5384615384615384], 'AppliedSciences Civics_Government': [0.44, 0.56], 'College_CareerPrep Health_LifeScience': [0.4, 0.6], 'Civics_Government EnvironmentalScience': [0.44, 0.56], 'Gym_Fitness PerformingArts': [0.48, 0.52], 'History_Geography PerformingArts': [0.44, 0.56], 'Health_Wellness History_Geography': [0.4, 0.6], 'CommunityService Extracurricular': [0.44, 0.56], 'EarlyDevelopment ParentInvolvement': [0.44, 0.56], 'Extracurricular Other': [0.44, 0.56], 'Parent Involvement': [0.44, 0.56], 'EnvironmentalScience NutritionEducation': [0.52, 0.48], 'Mathematics PerformingArts': [0.44, 0.56], 'CommunityService Literacy': [0.44, 0.56], 'CharacterEducation ESL': [0.4, 0.6], 'FinancialLiteracy Literacy': [0.44, 0.56], 'Health_LifeScience Other': [0.48, 0.52], 'PerformingArts TeamSports': [0.44, 0.56], 'Gym_Fitness Literacy': [0.48, 0.52], 'Extracurricular SpecialNeeds': [0.44, 0.56], 'EarlyDevelopment NutritionEducation': [0.4166666666666667, 0.5833333333333334], 'EarlyDevelopment Music': [0.4583333333333333, 0.5416666666666666], 'ParentInvolvement SpecialNeeds': [0.4583333333333333, 0.5416666666666666], 'College_CareerPrep Health_Wellness': [0.4583333333333333, 0.5416666666666666], 'AppliedSciences CommunityService': [0.4166666666666667, 0.5833333333333334], 'CharacterEducation ParentInvolvement': [0.4583333333333333, 0.5416666666666666], 'EarlyDevelopment PerformingArts': [0.4166666666666667, 0.5833333333333334], 'CommunityService EnvironmentalScience': [0.4166666666666667, 0.5833333333333334], 'Extracurricular Literature_Writing': [0.4583333333333333, 0.5416666666666666], 'ForeignLanguages Health_Wellness': [0.4583333333333333, 0.5416666666666666], 'ESL ParentInvolvement': [0.5, 0.5], 'College_CareerPrep CommunityService': [0.4166666666666667, 0.5833333333333334], 'EnvironmentalScience ParentInvolvement': [0.4166666666666667, 0.5833333333333334], 'CharacterEducation Health_LifeScience': [0.4166666666666667, 0.5833333333333334], 'College_CareerPrep ForeignLanguages': [0.4166666666666667, 0.5833333333333334], 'CharacterEducation TeamSports': [0.4583333333333333, 0.5416666666666666], 'Civics_Government VisualArts': [0.4166666666666667, 0.5833333333333334], 'Gym_Fitness Music': [0.4583333333333333, 0.5416666666666666], 'CommunityService VisualArts': [0.4583333333333333, 0.5416666666666666], 'Health_Wellness PerformingArts': [0.4166666666666667, 0.5833333333333334], 'Extracurricular ParentInvolvement': [0.4782608695652174, 0.5217391304347826], 'EarlyDevelopment TeamSports': [0.4782608695652174, 0.5217391304347826], 'ParentInvolvement SocialSciences': [0.43478260869565216, 0.5652173913043478], 'AppliedSciences ForeignLanguages': [0.5217391304347826, 0.4782608695652174], 'NutritionEducation Other': [0.4782608695652174, 0.5217391304347826], 'Gym_Fitness Literature_Writing': [0.4782608695652174, 0.5217391304347826], 'CommunityService Health_LifeScience': [0.43478260869565216, 0.5652173913043478], 'Civics_Government College_CareerPrep': [0.43478260869565216, 0.5652173913043478], 'Economics Literacy': [0.43478260869565216, 0.5652173913043478], 'PerformingArts SpecialNeeds':

[0.4782608695652174, 0.5217391304347826], 'CharacterEducation SocialSciences': [0.4782608695652174, 0.5217391304347826], 'College_CareerPrep Music': [0.5217391304347826, 0.4782608695652174], 'Music SocialSciences': [0.5217391304347826, 0.4782608695652174], 'Extracurricular Health_Wellness': [0.5217391304347826, 0.4782608695652174], 'CharacterEducation Music': [0.43478260869565216, 0.5652173913043478], 'Health_Wellness SocialSciences': [0.4782608695652174, 0.5217391304347826], 'Literature_Writing TeamSports': [0.4782608695652174, 0.5217391304347826], 'Gym_Fitness VisualArts': [0.4782608695652174, 0.5217391304347826], 'Extracurricular TeamSports': [0.43478260869565216, 0.5652173913043478], 'FinancialLiteracy History_Geography': [0.43478260869565216, 0.5652173913043478], 'ESL PerformingArts': [0.5217391304347826, 0.4782608695652174], 'CharacterEducation EnvironmentalScience': [0.43478260869565216, 0.5652173913043478], 'College_CareerPrep EarlyDevelopment': [0.43478260869565216, 0.5652173913043478], 'CommunityService Mathematics': [0.4782608695652174, 0.5217391304347826], 'Extracurricular PerformingArts': [0.45454545454545453, 0.5454545454545454], 'Civics_Government FinancialLiteracy': [0.5, 0.5], 'PerformingArts SocialSciences': [0.5, 0.5], 'EarlyDevelopment Extracurricular': [0.5, 0.5], 'Music ParentInvolvement': [0.45454545454545453, 0.5454545454545454], 'History_Geography Other': [0.5, 0.5], 'Literacy TeamSports': [0.45454545454545453, 0.5454545454545454], 'ForeignLanguages VisualArts': [0.45454545454545453, 0.5454545454545454], 'ForeignLanguages History_Geography': [0.45454545454545453, 0.5454545454545454], 'CommunityService Health_Wellness': [0.5, 0.5], 'Literature_Writing Warmth Care_Hunger': [0.45454545454545453, 0.5454545454545454], 'CommunityService ParentInvolvement': [0.45454545454545453, 0.5454545454545454], 'CharacterEducation Civics_Government': [0.45454545454545453, 0.5454545454545454], 'ESL SocialSciences': [0.5, 0.5], 'ForeignLanguages Music': [0.45454545454545453, 0.5454545454545454], 'Civics_Government Health_LifeScience': [0.45454545454545453, 0.5454545454545454], 'College_CareerPrep History_Geography': [0.5, 0.5], 'ParentInvolvement PerformingArts': [0.45454545454545453, 0.5454545454545454], 'Other ParentInvolvement': [0.45454545454545453, 0.5454545454545454], 'FinancialLiteracy Literature_Writing': [0.5, 0.5], 'Extracurricular Health_LifeScience': [0.45454545454545453, 0.5454545454545454], 'AppliedSciences TeamSports': [0.45454545454545453, 0.5454545454545454], 'Music TeamSports': [0.45454545454545453, 0.5454545454545454], 'EnvironmentalScience Extracurricular': [0.5, 0.5], 'CommunityService History_Geography': [0.45454545454545453, 0.5454545454545454], 'SpecialNeeds TeamSports': [0.5, 0.5], 'Mathematics TeamSports': [0.5454545454545454, 0.45454545454545453], 'EarlyDevelopment History_Geography': [0.45454545454545453, 0.5454545454545454], 'EarlyDevelopment Warmth Care_Hunger': [0.45454545454545453, 0.5454545454545454], 'Civics_Government Mathematics': [0.45454545454545453, 0.5454545454545454], 'College_CareerPrep Economics': [0.45454545454545453, 0.5454545454545454], 'AppliedSciences PerformingArts': [0.45454545454545453, 0.5454545454545454], 'Other PerformingArts': [0.5454545454545454, 0.45454545454545453], 'Health_LifeScience TeamSports': [0.45454545454545453, 0.5454545454545454], 'Economics Other': [0.47619047619047616, 0.5238095238095238], 'Civics_Government TeamSports': [0.47619047619047616, 0.5238095238095238], 'Health_LifeScience Warmth Care_Hunger': [0.47619047619047616, 0.5238095238095238], 'Gym_Fitness ParentInvolvement': [0.47619047619047616, 0.5238095238095238], 'FinancialLiteracy ParentInvolvement': [0.47619047619047616, 0.5238095238095238], 'Civics_Government SpecialNeeds': [0.47619047619047616, 0.5238095238095238], 'FinancialLiteracy Health_Wellness': [0.47619047619047616, 0.5238095238095238], 'Other Warmth Care_Hunger': [0.47619047619047616, 0.5238095238095238], 'EarlyDevelopment ForeignLanguages': [0.47619047619047616, 0.5238095238095238], 'CommunityService FinancialLiteracy': [0.47619047619047616, 0.5238095238095238], 'Gym_Fitness History_Geography': [0.47619047619047616, 0.5238095238095238], 'Extracurricular History_Geography': [0.47619047619047616, 0.5238095238095238], 'Economics EnvironmentalScience': [0.47619047619047616, 0.5238095238095238], 'CharacterEducation Gym_Fitness': [0.47619047619047616, 0.5238095238095238], 'EnvironmentalScience ForeignLanguages': [0.47619047619047616, 0.5238095238095238], 'EnvironmentalScience Gym_Fitness': [0.5238095238095238, 0.47619047619047616], 'ESL Music': [0.47619047619047616, 0.5238095238095238], 'EarlyDevelopment SocialSciences': [0.47619047619047616, 0.5238095238095238], 'Mathematics Warmth Care_Hunger': [0.5238095238095238, 0.47619047619047616], 'Civics_Government Health_Wellness': [0.47619047619047616, 0.5238095238095238], 'FinancialLiteracy SocialSciences': [0.47619047619047616, 0.5238095238095238], 'Extracurricular Gym_Fitness': [0.47619047619047616, 0.5238095238095238], 'AppliedSciences Gym_Fitness': [0.47619047619047616, 0.5238095238095238], 'CharacterEducation PerformingArts': [0.5238095238095238, 0.47619047619047616], 'College_CareerPrep NutritionEducation': [0.47619047619047616, 0.5238095238095238], 'CommunityService SocialSciences': [0.47619047619047616, 0.5238095238095238], 'Music Other': [0.47619047619047616, 0.5238095238095238], 'Health_Wellness ParentInvolvement': [0.47619047619047616, 0.5238095238095238], 'NutritionEducation SocialSciences': [0.47619047619047616, 0.5238095238095238], 'ESL Other': [0.47619047619047616, 0.5238095238095238], 'ForeignLanguages SpecialNeeds': [0.47619047619047616, 0.5238095238095238], 'SpecialNeeds Warmth Care_Hunger': [0.47619047619047616, 0.5238095238095238], 'EnvironmentalScience PerformingArts': [0.47619047619047616, 0.5238095238095238], 'Literacy NutritionEducation': [0.47619047619047616, 0.5238095238095238], 'Other SocialSciences': [0.47619047619047616, 0.5238095238095238], 'CommunityService Music': [0.47619047619047616, 0.5238095238095238], 'Economics VisualArts': [0.5238095238095238, 0.47619047619047616], 'ParentInvolvement TeamSports': [0.47619047619047616, 0.5238095238095238], 'Extracurricular NutritionEducation': [0.5238095238095238, 0.47619047619047616], 'EnvironmentalScience Music': [0.47619047619047616, 0.5238095238095238], 'CommunityService Economics': [0.47619047619047616, 0.5238095238095238], 'EnvironmentalScience Other': [0.5238095238095238, 0.47619047619047616], 'Economics Music': [0.47619047619047616, 0.5238095238095238], 'ForeignLanguages Other': [0.47619047619047616, 0.5238095238095238], 'ForeignLanguages PerformingArts': [0.47619047619047616, 0.5238095238095238], 'History_Geography ParentInvolvement': [0.47619047619047616, 0.5238095238095238], 'TeamSports VisualArts': [0.47619047619047616, 0.5238095238095238], 'ForeignLanguages Gym_Fitness': [0.47619047619047616, 0

.5238095238095238], 'Extracurricular SocialSciences': [0.47619047619047616, 0.5238095238095238], 'ForeignLanguages Health_LifeScience': [0.47619047619047616, 0.5238095238095238], 'Civics_Government ESL': [0.47619047619047616, 0.5238095238095238], 'CharacterEducation ForeignLanguages': [0.47619047619047616, 0.5238095238095238], 'Mathematics NutritionEducation': [0.47619047619047616, 0.5238095238095238], 'EarlyDevelopment FinancialLiteracy': [0.47619047619047616, 0.5238095238095238], 'Economics Literature_Writing': [0.47619047619047616, 0.5238095238095238], 'EnvironmentalScience TeamSports': [0.47619047619047616, 0.5238095238095238], 'Other TeamSports': [0.47619047619047616, 0.5238095238095238], 'VisualArts Warmth Care_Hunger': [0.5238095238095238, 0.47619047619047616], 'Health_LifeScience ParentInvolvement': [0.47619047619047616, 0.5238095238095238], 'CommunityService ESL': [0.47619047619047616, 0.5238095238095238], 'Gym_Fitness Health_LifeScience': [0.47619047619047616, 0.5238095238095238]}
{'Literacy': [0.13218970736629668, 0.8678102926337034], 'Literacy Mathematics': [0.13083048919226395, 0.8691695108077361], 'Literature_Writing Mathematics': [0.14677419354838708, 0.853225806451613], 'Literacy Literature_Writing': [0.16019417475728157, 0.8398058252427184], 'Mathematics': [0.18727272727272729, 0.8127272727272727], 'SpecialNeeds': [0.1772428884026258, 0.8227571115973742], 'Literature_Writing': [0.17792792792792791, 0.8220720720720721], 'Health_Wellness': [0.1524547803617571, 0.8475452196382429], 'AppliedSciences Mathematics': [0.19020172910662825, 0.8097982708933718], 'Literacy SpecialNeeds': [0.16312056737588654, 0.8368794326241135], 'AppliedSciences': [0.23193916349809887, 0.7680608365019012], 'Gym_Fitness Health_Wellness': [0.16015625, 0.83984375], 'ESL Literacy': [0.1630901287553648, 0.8369098712446352], 'VisualArts': [0.2290748898678414, 0.7709251101321586], 'Music': [0.1511627906976744, 0.8488372093023255], 'Warmth Care_Hunger': [0.1337579617834395, 0.8662420382165605], 'Health_Wellness SpecialNeeds': [0.1895424836601307, 0.8104575163398693], 'Literature_Writing SpecialNeeds': [0.2266666666666666, 0.7733333333333333], 'Mathematics SpecialNeeds': [0.24822695035460993, 0.75177304964539], 'Gym_Fitness': [0.2318840579710145, 0.7681159420289855], 'EnvironmentalScience': [0.232, 0.768], 'Music PerformingArts': [0.152, 0.848], 'AppliedSciences EnvironmentalScience': [0.25, 0.75], 'EnvironmentalScience Health_LifeScience': [0.2605042016806723, 0.7394957983193278], 'TeamSports': [0.23478260869565218, 0.7652173913043478], 'EnvironmentalScience Mathematics': [0.24324324324324326, 0.7567567567567568], 'Health_LifeScience': [0.2, 0.8], 'EarlyDevelopment': [0.24271844660194175, 0.7572815533980582], 'Health_Wellness NutritionEducation': [0.2, 0.8], 'EarlyDevelopment Literacy': [0.21, 0.79], 'Other': [0.2653061224489796, 0.7346938775510204], 'ESL Literature_Writing': [0.27956989247311825, 0.7204301075268817], 'AppliedSciences Health_LifeScience': [0.22727272727272727, 0.7727272727272727], 'History_Geography': [0.25882352941176473, 0.7411764705882353], 'Literacy VisualArts': [0.2987012987012987, 0.7012987012987013], 'AppliedSciences VisualArts': [0.23376623376623376, 0.7662337662337663], 'EarlyDevelopment SpecialNeeds': [0.23376623376623376, 0.7662337662337663], 'Literature_Writing VisualArts': [0.25, 0.75], 'Health_LifeScience Mathematics': [0.3108108108108108, 0.6891891891891891], 'History_Geography Literature_Writing': [0.1917808219178082, 0.8082191780821918], 'AppliedSciences Literacy': [0.25, 0.75], 'Mathematics VisualArts': [0.2857142857142857, 0.7142857142857143], 'History_Geography Literacy': [0.2028985507246377, 0.7971014492753623], 'Health_Wellness Literacy': [0.2753623188405797, 0.7246376811594203], 'PerformingArts': [0.23529411764705882, 0.7647058823529411], 'Gym_Fitness TeamSports': [0.27941176470588236, 0.72058823529411764], 'ESL': [0.22388059701492538, 0.7761194029850746], 'AppliedSciences Literature_Writing': [0.24242424242424243, 0.7575757575757576], 'College_CareerPrep': [0.2923076923076923, 0.7076923076923077], 'CharacterEducation Literacy': [0.25396825396825395, 0.746031746031746], 'EarlyDevelopment Health_Wellness': [0.22033898305084745, 0.7796610169491526], 'History_Geography SocialSciences': [0.22033898305084745, 0.7796610169491526], 'AppliedSciences College_CareerPrep': [0.288135593220339, 0.711864406779661], 'Literacy SocialSciences': [0.20689655172413793, 0.7931034482758621], 'EnvironmentalScience Literacy': [0.2857142857142857, 0.7142857142857143], 'College_CareerPrep Literature_Writing': [0.2909090909090909, 0.7090909090909091], 'Other SpecialNeeds': [0.2727272727272727, 0.7272727272727273], 'EarlyDevelopment Mathematics': [0.3454545454545454, 0.6545454545454545], 'EnvironmentalScience VisualArts': [0.3018867924528302, 0.6981132075471698], 'Health_Wellness TeamSports': [0.32075471698113206, 0.6792452830188679], 'Literature_Writing SocialSciences': [0.20754716981132076, 0.792452830188679], 'Health_LifeScience Literacy': [0.21153846153846154, 0.7884615384615384], 'Health_Wellness Mathematics': [0.36538461538461536, 0.6346153846153846], 'EnvironmentalScience Literature_Writing': [0.3137254901960784, 0.6862745098039216], 'SpecialNeeds VisualArts': [0.29411764705882354, 0.7058823529411765], 'AppliedSciences SpecialNeeds': [0.26, 0.74], 'CharacterEducation': [0.32, 0.68], 'Health_Wellness Literature_Writing': [0.2653061224489796, 0.7346938775510204], 'College_CareerPrep Mathematics': [0.2916666666666667, 0.7083333333333334], 'ForeignLanguages': [0.3541666666666667, 0.6458333333333334], 'ESL Mathematics': [0.2978723404255319, 0.7021276595744681], 'ForeignLanguages Literacy': [0.3191489361702128, 0.6808510638297872], 'CharacterEducation EarlyDevelopment': [0.3829787234042553, 0.6170212765957447], 'CharacterEducation Literature_Writing': [0.3695652173913043, 0.6304347826086957], 'Health_Wellness Other': [0.28888888888888886, 0.7111111111111111], 'NutritionEducation': [0.25, 0.75], 'AppliedSciences EarlyDevelopment': [0.3409090909090909, 0.6590909090909091], 'SocialSciences': [0.30952380952380953, 0.6904761904761905], 'Literacy ParentInvolvement': [0.3, 0.7], 'College_CareerPrep Literacy': [0.3, 0.7], 'Health_LifeScience Literature_Writing': [0.3076923076923077, 0.6923076923076923], 'History_Geography VisualArts': [0.34210526315789475, 0.6578947368421053], 'EnvironmentalScience SpecialNeeds': [0.42105263157894735, 0.5789473684210527], 'FinancialLiteracy': [0.3157894736842105, 0.6842105263157895], 'ESL SpecialNeeds': [0.2894736842105263, 0.7105263157894737], 'EarlyDevelopment Literature_Writing': [0.2972972972972973, 0.7027027027027027], 'EarlyDevelopment VisualArts': [0.3333333333333333, 0.6666666666666666], 'Civics_Government History_Geography': [0.3055555555555556, 0.6944444444444444], 'Civics_Government Literature_Writing': [0.3611111111111111, 0.6388888888888888], 'FinancialLiteracy Mathematics': [0.3055555555555556, 0.6944444444444444],

'EarlyDevelopment Other': [0.3888888888888889, 0.6111111111111112], 'Health_LifeScience Health_Wellness': [0.3611111111111111, 0.6388888888888888], 'Literacy Other': [0.34285714285714286, 0.6571428571428571], 'PerformingArts VisualArts': [0.4, 0.6], 'Music SpecialNeeds': [0.3142857142857143, 0.6857142857142857], 'CharacterEducation SpecialNeeds': [0.34285714285714286, 0.6571428571428571], 'College_CareerPrep SpecialNeeds': [0.38235294117647056, 0.6176470588235294], 'AppliedSciences ESL': [0.3235294117647059, 0.6764705882352942], 'Literature_Writing Other': [0.48484848484848486, 0.5151515151515151], 'EnvironmentalScience History_Geography': [0.36363636363636365, 0.6363636363636364], 'CharacterEducation Other': [0.3939393939393939, 0.6060606060606061], 'History_Geography SpecialNeeds': [0.3939393939393939, 0.6060606060606061], 'Gym_Fitness SpecialNeeds': [0.375, 0.625], 'History_Geography Mathematics': [0.46875, 0.53125], 'College_CareerPrep VisualArts': [0.34375, 0.65625], 'College_CareerPrep Other': [0.4375, 0.5625], 'Extracurricular': [0.40625, 0.59375], 'Civics_Government Literacy': [0.3125, 0.6875], 'CharacterEducation Health_Wellness': [0.3548387096774194, 0.6451612903225806], 'Literature_Writing PerformingArts': [0.3225806451612903, 0.6774193548387096], 'Health_LifeScience VisualArts': [0.41935483870967744, 0.5806451612903226], 'Health_LifeScience SpecialNeeds': [0.4, 0.6], 'ForeignLanguages Literature_Writing': [0.43333333333333335, 0.5666666666666667], 'Mathematics ParentInvolvement': [0.3333333333333333, 0.6666666666666666], 'AppliedSciences Extracurricular': [0.36666666666666664, 0.6333333333333333], 'CommunityService EnvironmentalScience': [0.3793103448275862, 0.6206896551724138], 'Extracurricular VisualArts': [0.3793103448275862, 0.6206896551724138], 'AppliedSciences Other': [0.3448275862068966, 0.6551724137931034], 'CharacterEducation Mathematics': [0.4482758620689655, 0.5517241379310345], 'Literacy Music': [0.41379310344827586, 0.5862068965517241], 'CommunityService VisualArts': [0.41379310344827586, 0.5862068965517241], 'Mathematics SocialSciences': [0.41379310344827586, 0.5862068965517241], 'Mathematics Other': [0.39285714285714285, 0.6071428571428571], 'Civics_Government SocialSciences': [0.39285714285714285, 0.6071428571428571], 'Health_LifeScience History_Geography': [0.42857142857142855, 0.5714285714285714], 'Civics_Government': [0.4642857142857143, 0.5357142857142857], 'Literacy PerformingArts': [0.35714285714285715, 0.6428571428571429], 'FinancialLiteracy SpecialNeeds': [0.48148148148148145, 0.5185185185185185], 'SocialSciences VisualArts': [0.37037037037037035, 0.6296296296296297], 'Extracurricular Mathematics': [0.4444444444444444, 0.5555555555555556], 'Health_LifeScience NutritionEducation': [0.37037037037037035, 0.6296296296296297], 'AppliedSciences ParentInvolvement': [0.4074074074074074, 0.5925925925925926], 'College_CareerPrep EarlyDevelopment': [0.37037037037037035, 0.6296296296296297], 'SocialSciences SpecialNeeds': [0.4444444444444444, 0.5555555555555556], 'AppliedSciences History_Geography': [0.37037037037037035, 0.6296296296296297], 'Gym_Fitness NutritionEducation': [0.37037037037037035, 0.6296296296296297], 'CharacterEducation VisualArts': [0.46153846153846156, 0.5384615384615384], 'CommunityService': [0.4230769230769231, 0.5769230769230769], 'CharacterEducation College_CareerPrep': [0.4230769230769231, 0.5769230769230769], 'AppliedSciences SocialSciences': [0.38461538461538464, 0.6153846153846154], 'Civics_Government SpecialNeeds': [0.5, 0.5], 'Mathematics Music': [0.4230769230769231, 0.5769230769230769], 'Economics': [0.38461538461538464, 0.6153846153846154], 'ForeignLanguages Mathematics': [0.44, 0.56], 'Health_Wellness Music': [0.44, 0.56], 'ESL VisualArts': [0.44, 0.56], 'EnvironmentalScience SocialSciences': [0.4, 0.6], 'CharacterEducation Extracurricular': [0.4, 0.6], 'Literature_Writing Music': [0.44, 0.56], 'EnvironmentalScience Health_Wellness': [0.44, 0.56], 'ParentInvolvement': [0.48, 0.52], 'College_CareerPrep Extracurricular': [0.44, 0.56], 'SpecialNeeds TeamSports': [0.44, 0.56], 'CharacterEducation EnvironmentalScience': [0.44, 0.56], 'ESL EnvironmentalScience': [0.4, 0.6], 'ESL EarlyDevelopment': [0.4, 0.6], 'SpecialNeeds Warmth Care_Hunger': [0.44, 0.56], 'CharacterEducation Music': [0.44, 0.56], 'College_CareerPrep Health_LifeScience': [0.48, 0.52], 'Music VisualArts': [0.48, 0.52], 'College_CareerPrep ParentInvolvement': [0.44, 0.56], 'Gym_Fitness Mathematics': [0.44, 0.56], 'AppliedSciences Music': [0.44, 0.56], 'EnvironmentalScience NutritionEducation': [0.44, 0.56], 'Extracurricular Literature_Writing': [0.44, 0.56], 'ESL ForeignLanguages': [0.5416666666666666, 0.4583333333333333], 'Extracurricular PerformingArts': [0.4166666666666667, 0.5833333333333334], 'Extracurricular SpecialNeeds': [0.4583333333333333, 0.5416666666666666], 'AppliedSciences CharacterEducation': [0.4583333333333333, 0.5416666666666666], 'Literacy TeamSports': [0.4583333333333333, 0.5416666666666666], 'AppliedSciences Health_Wellness': [0.4583333333333333, 0.5416666666666666], 'CharacterEducation ParentInvolvement': [0.4583333333333333, 0.5416666666666666], 'AppliedSciences PerformingArts': [0.4166666666666667, 0.5833333333333334], 'Health_Wellness SocialSciences': [0.4583333333333333, 0.5416666666666666], 'Health_LifeScience SocialSciences': [0.5416666666666666, 0.4583333333333333], 'Economics FinancialLiteracy': [0.5416666666666666, 0.4583333333333333], 'Gym_Fitness Literacy': [0.5217391304347826, 0.4782608695652174], 'Economics Mathematics': [0.43478260869565216, 0.5652173913043478], 'ESL Health_Wellness': [0.4782608695652174, 0.5217391304347826], 'ESL SocialSciences': [0.5217391304347826, 0.4782608695652174], 'College_CareerPrep CommunityService': [0.43478260869565216, 0.5652173913043478], 'ESL History_Geography': [0.43478260869565216, 0.5652173913043478], 'Civics_Government Economics': [0.43478260869565216, 0.5652173913043478], 'ParentInvolvement SpecialNeeds': [0.4782608695652174, 0.5217391304347826], 'College_CareerPrep Health_Wellness': [0.4782608695652174, 0.5217391304347826], 'Health_Wellness VisualArts': [0.5217391304347826, 0.4782608695652174], 'Literature_Writing ParentInvolvement': [0.43478260869565216, 0.5652173913043478], 'AppliedSciences Gym_Fitness': [0.43478260869565216, 0.5652173913043478], 'EarlyDevelopment Health_LifeScience': [0.43478260869565216, 0.5652173913043478], 'ForeignLanguages History_Geography': [0.43478260869565216, 0.5652173913043478], 'ForeignLanguages SpecialNeeds': [0.43478260869565216, 0.5652173913043478], 'EarlyDevelopment Gym_Fitness': [0.4782608695652174, 0.5217391304347826], 'CharacterEducation CommunityService': [0.5217391304347826, 0.4782608695652174], 'Gym_Fitness Other': [0.4782608695652174, 0.5217391304347826], 'FinancialLiteracy Literacy': [0.43478260869565216, 0.5652173913043478], 'College_CareerPrep History_Geography': [0.43478260869565216, 0.5652173913043478], 'NutritionEducation SpecialNeeds':

[0.4782608695652174, 0.5217391304347826], 'EarlyDevelopment ParentInvolvement':
[0.43478260869565216, 0.5652173913043478], 'CommunityService Extracurricular':
[0.4782608695652174, 0.5217391304347826], 'Extracurricular Other': [0.4782608695652174,
0.5217391304347826], 'Extracurricular Literacy': [0.4782608695652174, 0.5217391304347826],
'College_CareerPrep SocialSciences': [0.43478260869565216, 0.5652173913043478], 'Extracurricular H
istory_Geography': [0.45454545454545453, 0.5454545454545454], 'EarlyDevelopment
EnvironmentalScience': [0.5454545454545454, 0.45454545454545453], 'CommunityService Literacy': [0.
5454545454545454, 0.45454545454545453], 'Civics_Government VisualArts': [0.45454545454545453, 0.54
545454545454545], 'CharacterEducation History_Geography': [0.45454545454545453,
0.5454545454545454], 'AppliedSciences NutritionEducation': [0.5, 0.5], 'Other VisualArts':
[0.45454545454545453, 0.5454545454545454], 'College_CareerPrep Music': [0.5, 0.5],
'EarlyDevelopment Extracurricular': [0.45454545454545453, 0.5454545454545454], 'College_CareerPrep
NutritionEducation': [0.5, 0.5], 'CharacterEducation TeamSports': [0.5454545454545454,
0.45454545454545453], 'Gym_Fitness Health_LifeScience': [0.45454545454545453, 0.5454545454545454],
'Civics_Government Health_LifeScience': [0.5, 0.5], 'Mathematics PerformingArts':
[0.45454545454545453, 0.5454545454545454], 'Health_LifeScience Music': [0.45454545454545453,
0.5454545454545454], 'EnvironmentalScience ForeignLanguages': [0.45454545454545453,
0.5454545454545454], 'Civics_Government Mathematics': [0.45454545454545453, 0.5454545454545454], '
ParentInvolvement VisualArts': [0.45454545454545453, 0.5454545454545454], 'EarlyDevelopment
SocialSciences': [0.5, 0.5], 'Economics SocialSciences': [0.45454545454545453,
0.5454545454545454], 'CharacterEducation Gym_Fitness': [0.5, 0.5], 'Economics Literacy':
[0.45454545454545453, 0.5454545454545454], 'Civics_Government CommunityService': [0.5, 0.5],
'Civics_Government College_CareerPrep': [0.5, 0.5], 'ESL Health_LifeScience':
[0.45454545454545453, 0.5454545454545454], 'AppliedSciences CommunityService': [0.5, 0.5],
'EnvironmentalScience Other': [0.45454545454545453, 0.5454545454545454], 'PerformingArts
SpecialNeeds': [0.5, 0.5], 'Extracurricular Music': [0.45454545454545453, 0.5454545454545454],
'Health_Wellness Warmth Care_Hunger': [0.45454545454545453, 0.5454545454545454], 'Music
SocialSciences': [0.45454545454545453, 0.5454545454545454], 'CommunityService Health_Wellness': [0
.45454545454545453, 0.5454545454545454], 'Civics_Government TeamSports': [0.47619047619047616,
0.5238095238095238], 'Other TeamSports': [0.47619047619047616, 0.5238095238095238],
'CharacterEducation Warmth Care_Hunger': [0.5238095238095238, 0.47619047619047616],
'EnvironmentalScience Music': [0.47619047619047616, 0.5238095238095238], 'Civics_Government
EnvironmentalScience': [0.47619047619047616, 0.5238095238095238], 'College_CareerPrep
FinancialLiteracy': [0.47619047619047616, 0.5238095238095238], 'Gym_Fitness SocialSciences':
[0.47619047619047616, 0.5238095238095238], 'Civics_Government Extracurricular':
[0.5238095238095238, 0.47619047619047616], 'Extracurricular Gym_Fitness': [0.47619047619047616, 0.
5238095238095238], 'CommunityService Other': [0.47619047619047616, 0.5238095238095238],
'CommunityService Mathematics': [0.5238095238095238, 0.47619047619047616], 'CommunityService ESL':
[0.5238095238095238, 0.47619047619047616], 'CommunityService PerformingArts':
[0.47619047619047616, 0.5238095238095238], 'Health_LifeScience Other': [0.47619047619047616,
0.5238095238095238], 'CharacterEducation FinancialLiteracy': [0.47619047619047616,
0.5238095238095238], 'CommunityService Literature_Writing': [0.47619047619047616,
0.5238095238095238], 'CommunityService Economics': [0.47619047619047616, 0.5238095238095238], 'Mat
hematics Warmth Care_Hunger': [0.47619047619047616, 0.5238095238095238], 'Extracurricular
SocialSciences': [0.47619047619047616, 0.5238095238095238], 'CharacterEducation
Health_LifeScience': [0.47619047619047616, 0.5238095238095238], 'Gym_Fitness VisualArts':
[0.47619047619047616, 0.5238095238095238], 'CharacterEducation ForeignLanguages':
[0.47619047619047616, 0.5238095238095238], 'Other ParentInvolvement': [0.47619047619047616,
0.5238095238095238], 'ParentInvolvement PerformingArts': [0.47619047619047616,
0.5238095238095238], 'CharacterEducation PerformingArts': [0.47619047619047616,
0.5238095238095238], 'EarlyDevelopment Music': [0.47619047619047616, 0.5238095238095238], 'ESL Par
entInvolvement': [0.47619047619047616, 0.5238095238095238], 'NutritionEducation VisualArts':
[0.47619047619047616, 0.5238095238095238], 'EnvironmentalScience FinancialLiteracy':
[0.47619047619047616, 0.5238095238095238], 'Civics_Government Health_Wellness':
[0.47619047619047616, 0.5238095238095238], 'FinancialLiteracy VisualArts': [0.47619047619047616, 0
.5238095238095238], 'AppliedSciences Civics_Government': [0.47619047619047616,
0.5238095238095238], 'EnvironmentalScience ParentInvolvement': [0.47619047619047616,
0.5238095238095238], 'Gym_Fitness Music': [0.47619047619047616, 0.5238095238095238],
'AppliedSciences FinancialLiteracy': [0.5238095238095238, 0.47619047619047616], 'Economics
EnvironmentalScience': [0.47619047619047616, 0.5238095238095238], 'Gym_Fitness
Literature_Writing': [0.47619047619047616, 0.5238095238095238], 'College_CareerPrep TeamSports': [
0.47619047619047616, 0.5238095238095238], 'CharacterEducation ESL': [0.5238095238095238,
0.47619047619047616], 'College_CareerPrep EnvironmentalScience': [0.47619047619047616,
0.5238095238095238], 'EnvironmentalScience Warmth Care_Hunger': [0.47619047619047616,
0.5238095238095238], 'Health_LifeScience ParentInvolvement': [0.47619047619047616,
0.5238095238095238], 'CommunityService NutritionEducation': [0.5238095238095238,
0.47619047619047616], 'PerformingArts SocialSciences': [0.47619047619047616, 0.5238095238095238],
'Extracurricular Health_Wellness': [0.47619047619047616, 0.5238095238095238], 'ParentInvolvement W
armth Care_Hunger': [0.47619047619047616, 0.5238095238095238], 'ForeignLanguages
Health_LifeScience': [0.47619047619047616, 0.5238095238095238], 'Economics VisualArts':
[0.47619047619047616, 0.5238095238095238], 'EarlyDevelopment ForeignLanguages':
[0.47619047619047616, 0.5238095238095238], 'Gym_Fitness History_Geography': [0.47619047619047616,
0.5238095238095238], 'CommunityService SpecialNeeds': [0.5238095238095238, 0.47619047619047616], '
Extracurricular ForeignLanguages': [0.47619047619047616, 0.5238095238095238], 'ForeignLanguages Mu
sic': [0.47619047619047616, 0.5238095238095238], 'Civics_Government FinancialLiteracy':
[0.47619047619047616, 0.5238095238095238], 'Music TeamSports': [0.5238095238095238,
0.47619047619047616], 'NutritionEducation SocialSciences': [0.47619047619047616,

0.5238095238095238], 'College_CareerPrep PerformingArts': [0.5238095238095238,
0.47619047619047616], 'EarlyDevelopment Warmth Care_Hunger': [0.47619047619047616,
0.5238095238095238], 'History_Geography PerformingArts': [0.47619047619047616,
0.5238095238095238], 'College_CareerPrep ForeignLanguages': [0.47619047619047616,
0.5238095238095238], 'History_Geography Other': [0.47619047619047616, 0.5238095238095238],
'CommunityService EarlyDevelopment': [0.47619047619047616, 0.5238095238095238], 'AppliedSciences T
eamSports': [0.47619047619047616, 0.5238095238095238], 'FinancialLiteracy History_Geography':
[0.47619047619047616, 0.5238095238095238], 'AppliedSciences Economics': [0.5238095238095238,
0.47619047619047616], 'Extracurricular TeamSports': [0.47619047619047616, 0.5238095238095238], 'Oth
er SocialSciences': [0.47619047619047616, 0.5238095238095238]}
{'Literacy': [0.11029411764705882, 0.8897058823529411], 'Literacy Mathematics':
[0.12924071082390953, 0.8707592891760905], 'Literature_Writing Mathematics': [0.17349137931034483,
0.8265086206896551], 'Literacy Literature_Writing': [0.1490104772991851, 0.8509895227008148],
'Mathematics': [0.19813519813519814, 0.8018648018648019], 'Literature_Writing':
[0.1577424023154848, 0.8422575976845152], 'SpecialNeeds': [0.21348314606741572,
0.7865168539325843], 'Health_Wellness': [0.140597539543058, 0.859402460456942], 'AppliedSciences M
athematics': [0.17764471057884232, 0.8223552894211577], 'AppliedSciences': [0.22388059701492538, 0
.7761194029850746], 'Literacy SpecialNeeds': [0.16062176165803108, 0.8393782383419689], 'ESL
Literacy': [0.1394736842105263, 0.8605263157894737], 'VisualArts': [0.18681318681318682,
0.8131868131868132], 'Gym_Fitness Health_Wellness': [0.14814814814814814, 0.8518518518518519], 'Mu
sic': [0.1440677966101695, 0.8559322033898306], 'Gym_Fitness': [0.23148148148148148,
0.7685185185185185], 'Warmth Care_Hunger': [0.11267605633802817, 0.8873239436619719],
'Literature_Writing SpecialNeeds': [0.18660287081339713, 0.8133971291866029], 'Health_Wellness Spe
cialNeeds': [0.14705882352941177, 0.8529411764705882], 'Mathematics SpecialNeeds':
[0.2512820512820513, 0.7487179487179487], 'TeamSports': [0.27807486631016043, 0.7219251336898396],
'EnvironmentalScience': [0.1989247311827957, 0.80107526881720431], 'AppliedSciences
EnvironmentalScience': [0.2411764705882353, 0.7588235294117647], 'EnvironmentalScience
Mathematics': [0.22560975609756098, 0.774390243902439], 'EarlyDevelopment': [0.2331288343558282, 0
.7668711656441718], 'EnvironmentalScience Health_LifeScience': [0.21739130434782608,
0.782608695652174], 'Music PerformingArts': [0.14465408805031446, 0.8553459119496856],
'Health_LifeScience': [0.22580645161290322, 0.7741935483870968], 'Health_Wellness
NutritionEducation': [0.18055555555555555, 0.8194444444444444], 'Other': [0.2536231884057971, 0.74
63768115942029], 'ESL Literature_Writing': [0.20437956204379562, 0.7956204379562044],
'EarlyDevelopment SpecialNeeds': [0.2222222222222222, 0.7777777777777778], 'EarlyDevelopment
Literacy': [0.2204724409448819, 0.7795275590551181], 'Literature_Writing VisualArts':
[0.24603174603174602, 0.753968253968254], 'Gym_Fitness TeamSports': [0.2661290322580645,
0.7338709677419355], 'Literacy VisualArts': [0.21739130434782608, 0.782608695652174],
'History_Geography Literature_Writing': [0.14912280701754385, 0.8508771929824561],
'AppliedSciences VisualArts': [0.26605504587155965, 0.7339449541284404], 'AppliedSciences
Literacy': [0.17757009345794392, 0.822429906542056], 'History_Geography': [0.3018867924528302, 0.6
9811132075471698], 'Health_LifeScience Mathematics': [0.1792452830188679, 0.8207547169811321],
'History_Geography Literacy': [0.125, 0.875], 'EnvironmentalScience Literacy':
[0.21649484536082475, 0.7835051546391752], 'AppliedSciences Health_LifeScience':
[0.21649484536082475, 0.7835051546391752], 'Mathematics VisualArts': [0.2604166666666667,
0.7395833333333334], 'Health_Wellness Literacy': [0.2222222222222222, 0.7777777777777778],
'PerformingArts': [0.21348314606741572, 0.7865168539325843], 'ESL': [0.2159090909090909,
0.7840909090909091], 'AppliedSciences College_CareerPrep': [0.20689655172413793,
0.7931034482758621], 'AppliedSciences SpecialNeeds': [0.20930232558139536, 0.7906976744186046], 'C
haracterEducation': [0.27710843373493976, 0.7228915662650602], 'Health_Wellness TeamSports':
[0.2222222222222222, 0.7777777777777778], 'College_CareerPrep': [0.24358974358974358,
0.7564102564102564], 'Other SpecialNeeds': [0.24675324675324675, 0.7532467532467533],
'ForeignLanguages': [0.3157894736842105, 0.6842105263157895], 'AppliedSciences
Literature_Writing': [0.22972972972972974, 0.7702702702702703], 'College_CareerPrep Mathematics':
[0.273972602739726, 0.726027397260274], 'Literacy SocialSciences': [0.2328767123287671,
0.7671232876712328], 'College_CareerPrep Literature_Writing': [0.20833333333333334,
0.7916666666666666], 'EarlyDevelopment Mathematics': [0.2857142857142857, 0.7142857142857143],
'History_Geography SocialSciences': [0.2608695652173913, 0.7391304347826086], 'CharacterEducation
Literacy': [0.2153846153846154, 0.7846153846153846], 'NutritionEducation': [0.3076923076923077, 0.
6923076923076923], 'EnvironmentalScience Literature_Writing': [0.2153846153846154,
0.7846153846153846], 'SpecialNeeds VisualArts': [0.1875, 0.8125], 'Health_LifeScience Literacy': [
0.25, 0.75], 'Literature_Writing SocialSciences': [0.25806451612903225, 0.7419354838709677], 'Coll
ege_CareerPrep Literacy': [0.23214285714285715, 0.7678571428571429], 'Health_Wellness
Literature_Writing': [0.26785714285714285, 0.7321428571428571], 'ForeignLanguages Literacy':
[0.25925925925925924, 0.7407407407407407], 'ESL SpecialNeeds': [0.33962264150943394,
0.6603773584905660], 'EarlyDevelopment Health_Wellness': [0.2884615384615384, 0.7115384615384616],
'ESL Mathematics': [0.2692307692307692, 0.7307692307692307], 'Civics_Government
History_Geography': [0.2692307692307692, 0.7307692307692307], 'Health_Wellness Mathematics':
[0.28, 0.72], 'Health_LifeScience Literature_Writing': [0.2653061224489796, 0.7346938775510204], '
SocialSciences': [0.2857142857142857, 0.7142857142857143], 'Literacy Other': [0.3469387755102041,
0.6530612244897959], 'EarlyDevelopment Other': [0.32653061224489793, 0.6734693877551020],
'EarlyDevelopment Literature_Writing': [0.3333333333333333, 0.6666666666666666],
'College_CareerPrep VisualArts': [0.2916666666666667, 0.7083333333333334], 'EnvironmentalScience S
pecialNeeds': [0.2916666666666667, 0.7083333333333334], 'EnvironmentalScience VisualArts':
[0.2708333333333333, 0.7291666666666666], 'College_CareerPrep SpecialNeeds': [0.3333333333333333,
0.6666666666666666], 'Health_Wellness Other': [0.2765957446808511, 0.7234042553191489],
'Literature_Writing Other': [0.28888888888888886, 0.7111111111111111], 'Health_LifeScience
SpecialNeeds': [0.37777777777777777, 0.6222222222222222], 'CharacterEducation SpecialNeeds': [0.4,

0.6], 'FinancialLiteracy': [0.3488372093023256, 0.6511627906976745], 'CharacterEducation Literature_Writing': [0.32558139534883723, 0.6744186046511628], 'AppliedSciences EarlyDevelopment': [0.3488372093023256, 0.6511627906976745], 'Gym_Fitness SpecialNeeds': [0.3023255813953488, 0.6976744186046512], 'Literacy Music': [0.2619047619047619, 0.7380952380952381], 'EnvironmentalScience History_Geography': [0.3333333333333333, 0.6666666666666666], 'College_CareerPrep Other': [0.3170731707317073, 0.6829268292682927], 'Mathematics Other': [0.3902439024390244, 0.6097560975609756], 'Health_LifeScience Health_Wellness': [0.3170731707317073, 0.6829268292682927], 'CharacterEducation College_CareerPrep': [0.3, 0.7], 'Civics_Government Literacy': [0.3, 0.7], 'AppliedSciences Extracurricular': [0.28205128205128205, 0.717948717948718], 'Extracurricular': [0.358974358974359, 0.6410256410256411], 'Literature_Writing PerformingArts': [0.38461538461538464, 0.6153846153846154], 'Literacy ParentInvolvement': [0.358974358974359, 0.6410256410256411], 'History_Geography VisualArts': [0.28205128205128205, 0.717948717948718], 'CharacterEducation EarlyDevelopment': [0.3333333333333333, 0.6666666666666666], 'History_Geography Mathematics': [0.358974358974359, 0.6410256410256411], 'AppliedSciences Other': [0.39473684210526316, 0.6052631578947368], 'EarlyDevelopment VisualArts': [0.40540540540540543, 0.5945945945945946], 'Civics_Government SocialSciences': [0.2972972972972973, 0.7027027027027027], 'FinancialLiteracy Mathematics': [0.3055555555555556, 0.6944444444444444], 'Music SpecialNeeds': [0.3055555555555556, 0.6944444444444444], 'CharacterEducation Other': [0.34285714285714286, 0.6571428571428571], 'EnvironmentalScience SocialSciences': [0.4, 0.6], 'Health_LifeScience SocialSciences': [0.3142857142857143, 0.6857142857142857], 'Literacy PerformingArts': [0.3333333333333333, 0.6666666666666666], 'Civics_Government': [0.3939393939393939, 0.6060606060606061], 'CharacterEducation Health_Wellness': [0.36363636363636365, 0.6363636363636364], 'Extracurricular VisualArts': [0.3939393939393939, 0.6060606060606061], 'PerformingArts VisualArts': [0.46875, 0.53125], 'CharacterEducation CommunityService': [0.40625, 0.59375], 'History_Geography SpecialNeeds': [0.4375, 0.5625], 'SocialSciences VisualArts': [0.34375, 0.65625], 'CharacterEducation VisualArts': [0.34375, 0.65625], 'AppliedSciences History_Geography': [0.4375, 0.5625], 'Gym_Fitness NutritionEducation': [0.375, 0.625], 'CharacterEducation ParentInvolvement': [0.3870967741935484, 0.6129032258064516], 'Economics FinancialLiteracy': [0.41935483870967744, 0.5806451612903226], 'ForeignLanguages Literature_Writing': [0.3548387096774194, 0.6451612903225806], 'CommunityService': [0.41935483870967744, 0.5806451612903226], 'Health_LifeScience VisualArts': [0.3548387096774194, 0.6451612903225806], 'Literature_Writing ParentInvolvement': [0.3548387096774194, 0.6451612903225806], 'CharacterEducation Mathematics': [0.3548387096774194, 0.6451612903225806], 'Mathematics SocialSciences': [0.41935483870967744, 0.5806451612903226], 'AppliedSciences ESL': [0.4, 0.6], 'Extracurricular Literacy': [0.43333333333333335, 0.5666666666666667], 'Mathematics ParentInvolvement': [0.4, 0.6], 'Other VisualArts': [0.43333333333333335, 0.5666666666666667], 'College_CareerPrep Health_LifeScience': [0.3448275862068966, 0.6551724137931034], 'Music VisualArts': [0.3793103448275862, 0.6206896551724138], 'EarlyDevelopment ParentInvolvement': [0.41379310344827586, 0.5862068965517241], 'ESL EarlyDevelopment': [0.3793103448275862, 0.6206896551724138], 'College_CareerPrep Extracurricular': [0.39285714285714285, 0.6071428571428571], 'CommunityService EnvironmentalScience': [0.42857142857142855, 0.5714285714285714], 'Civics_Government Literature_Writing': [0.35714285714285715, 0.6428571428571429], 'Health_Wellness Music': [0.42857142857142855, 0.5714285714285714], 'ParentInvolvement VisualArts': [0.39285714285714285, 0.6071428571428571], 'CharacterEducation Extracurricular': [0.35714285714285715, 0.6428571428571429], 'Literature_Writing Music': [0.4074074074074074, 0.5925925925925926], 'Extracurricular Mathematics': [0.4074074074074074, 0.5925925925925926], 'SpecialNeeds TeamSports': [0.4444444444444444, 0.5555555555555556], 'Health_LifeScience History_Geography': [0.37037037037037035, 0.6296296296296297], 'ESL History_Geography': [0.4074074074074074, 0.5925925925925926], 'Extracurricular Other': [0.4074074074074074, 0.5925925925925926], 'AppliedSciences ParentInvolvement': [0.4074074074074074, 0.5925925925925926], 'College_CareerPrep PerformingArts': [0.4444444444444444, 0.5555555555555556], 'Health_LifeScience NutritionEducation': [0.5384615384615384, 0.46153846153846156], 'Extracurricular Literature_Writing': [0.38461538461538464, 0.6153846153846154], 'Extracurricular PerformingArts': [0.38461538461538464, 0.6153846153846154], 'Mathematics Music': [0.38461538461538464, 0.6153846153846154], 'AppliedSciences SocialSciences': [0.4230769230769231, 0.5769230769230769], 'Health_Wellness VisualArts': [0.4230769230769231, 0.5769230769230769], 'AppliedSciences PerformingArts': [0.4230769230769231, 0.5769230769230769], 'Economics': [0.4230769230769231, 0.5769230769230769], 'AppliedSciences Music': [0.4230769230769231, 0.5769230769230769], 'Economics History_Geography': [0.4230769230769231, 0.5769230769230769], 'SocialSciences SpecialNeeds': [0.4230769230769231, 0.5769230769230769], 'EnvironmentalScience NutritionEducation': [0.52, 0.48], 'Extracurricular TeamSports': [0.4, 0.6], 'ForeignLanguages VisualArts': [0.48, 0.52], 'ESL Health_LifeScience': [0.4, 0.6], 'Economics Mathematics': [0.4, 0.6], 'CharacterEducation SocialSciences': [0.44, 0.56], 'AppliedSciences Gym_Fitness': [0.48, 0.52], 'CommunityService Extracurricular': [0.44, 0.56], 'PerformingArts SpecialNeeds': [0.4, 0.6], 'CommunityService VisualArts': [0.48, 0.52], 'EnvironmentalScience Health_Wellness': [0.48, 0.52], 'EarlyDevelopment EnvironmentalScience': [0.44, 0.56], 'College_CareerPrep ParentInvolvement': [0.4583333333333333, 0.5416666666666666], 'EnvironmentalScience ParentInvolvement': [0.4166666666666667, 0.5833333333333334], 'NutritionEducation SpecialNeeds': [0.4583333333333333, 0.5416666666666666], 'ForeignLanguages History_Geography': [0.4166666666666667, 0.5833333333333334], 'EarlyDevelopment PerformingArts': [0.4166666666666667, 0.5833333333333334], 'ForeignLanguages Mathematics': [0.5, 0.5], 'Health_Wellness Warmth Care_Hunger': [0.4166666666666667, 0.5833333333333334], 'College_CareerPrep EnvironmentalScience': [0.4166666666666667, 0.5833333333333334], 'Health_LifeScience Music': [0.4166666666666667, 0.5833333333333334], 'Extracurricular Music': [0.5, 0.5], 'Health_Wellness History_Geography': [0.4166666666666667, 0.5833333333333334], 'NutritionEducation TeamSports': [0.4583333333333333, 0.5416666666666666], 'EarlyDevelopment Health_LifeScience': [0.4166666666666667, 0.5833333333333334], 'CommunityService SpecialNeeds': [0

Health_LifeScience': [0.41666666666666667, 0.5833333333333334], 'CommunityService SpecialNeeds': [0
.4166666666666667, 0.5833333333333334], 'ESL VisualArts': [0.4166666666666667,
0.5833333333333334], 'College_CareerPrep EarlyDevelopment': [0.4166666666666667,
0.5833333333333334], 'Civics_Government FinancialLiteracy': [0.4166666666666667,
0.5833333333333334], 'Extracurricular Health_Wellness': [0.4166666666666667, 0.5833333333333334],
'Gym_Fitness Mathematics': [0.5, 0.5], 'ParentInvolvement': [0.5, 0.5], 'Civics_Government
Economics': [0.4583333333333333, 0.5416666666666666], 'Gym_Fitness Literacy': [0.4583333333333333,
0.5416666666666666], 'CommunityService Literature_Writing': [0.4583333333333333,
0.5416666666666666], 'AppliedSciences CommunityService': [0.4583333333333333, 0.5416666666666666],
'Civics_Government Health_LifeScience': [0.4166666666666667, 0.5833333333333334], 'ESL
ForeignLanguages': [0.4166666666666667, 0.5833333333333334], 'CommunityService Mathematics':
[0.4166666666666667, 0.5833333333333334], 'CharacterEducation Health_LifeScience': [0.5, 0.5],
'ESL EnvironmentalScience': [0.4583333333333333, 0.5416666666666666], 'Economics SocialSciences':
[0.43478260869565216, 0.5652173913043478], 'College_CareerPrep CommunityService':
[0.4782608695652174, 0.5217391304347826], 'Civics_Government SpecialNeeds': [0.43478260869565216,
0.5652173913043478], 'CommunityService Health_Wellness': [0.4782608695652174, 0.5217391304347826],
'ESL PerformingArts': [0.43478260869565216, 0.5652173913043478], 'EarlyDevelopment Gym_Fitness': [
0.43478260869565216, 0.5652173913043478], 'EarlyDevelopment Music': [0.43478260869565216,
0.5652173913043478], 'ForeignLanguages SocialSciences': [0.4782608695652174, 0.5217391304347826],
'CharacterEducation TeamSports': [0.43478260869565216, 0.5652173913043478], 'AppliedSciences
CharacterEducation': [0.4782608695652174, 0.5217391304347826], 'Mathematics PerformingArts':
[0.4782608695652174, 0.5217391304347826], 'Health_Wellness PerformingArts': [0.4782608695652174, 0
.5217391304347826], 'Gym_Fitness Music': [0.4782608695652174, 0.5217391304347826],
'History_Geography Music': [0.43478260869565216, 0.5652173913043478], 'History_Geography
PerformingArts': [0.43478260869565216, 0.5652173913043478], 'College_CareerPrep Health_Wellness':
[0.4782608695652174, 0.5217391304347826], 'Health_Wellness SocialSciences': [0.43478260869565216,
0.5652173913043478], 'AppliedSciences TeamSports': [0.43478260869565216, 0.5652173913043478], 'ESL
Gym_Fitness': [0.5, 0.5], 'College_CareerPrep FinancialLiteracy': [0.5454545454545454,
0.45454545454545453], 'AppliedSciences Health_Wellness': [0.45454545454545453,
0.5454545454545454], 'Gym_Fitness Other': [0.45454545454545453, 0.5454545454545454],
'FinancialLiteracy Literacy': [0.45454545454545453, 0.5454545454545454], 'Other
ParentInvolvement': [0.5, 0.5], 'College_CareerPrep ESL': [0.45454545454545453,
0.5454545454545454], 'CommunityService History_Geography': [0.45454545454545453,
0.5454545454545454], 'Mathematics TeamSports': [0.45454545454545453, 0.5454545454545454],
'ForeignLanguages SpecialNeeds': [0.45454545454545453, 0.5454545454545454], 'College_CareerPrep So
cialSciences': [0.45454545454545453, 0.5454545454545454], 'College_CareerPrep NutritionEducation':
[0.5, 0.5], 'NutritionEducation Other': [0.45454545454545453, 0.5454545454545454],
'Civics_Government EnvironmentalScience': [0.5, 0.5], 'CharacterEducation Gym_Fitness': [0.5,
0.5], 'Other TeamSports': [0.45454545454545453, 0.5454545454545454], 'CharacterEducation
PerformingArts': [0.45454545454545453, 0.5454545454545454], 'Civics_Government Mathematics':
[0.5454545454545454, 0.45454545454545453], 'Economics SpecialNeeds': [0.5, 0.5],
'College_CareerPrep ForeignLanguages': [0.5454545454545454, 0.45454545454545453], 'ESL Music':
[0.45454545454545453, 0.5454545454545454], 'ESL Extracurricular': [0.45454545454545453,
0.5454545454545454], 'CharacterEducation ForeignLanguages': [0.5, 0.5], 'Gym_Fitness
PerformingArts': [0.45454545454545453, 0.5454545454545454], 'CharacterEducation
EnvironmentalScience': [0.5, 0.5], 'Extracurricular Health_LifeScience': [0.5454545454545454,
0.45454545454545453], 'Literature_Writing TeamSports': [0.45454545454545453, 0.5454545454545454],
'EnvironmentalScience Extracurricular': [0.5, 0.5], 'ParentInvolvement SpecialNeeds': [0.5, 0.5],
'ParentInvolvement SocialSciences': [0.5, 0.5], 'CharacterEducation Music': [0.5, 0.5],
'FinancialLiteracy SpecialNeeds': [0.45454545454545453, 0.5454545454545454], 'EarlyDevelopment Ext
racurricular': [0.5, 0.5], 'ESL Other': [0.5, 0.5], 'Civics_Government VisualArts': [0.545454545454
5454, 0.45454545454545453], 'EarlyDevelopment SocialSciences': [0.45454545454545453,
0.5454545454545454], 'FinancialLiteracy Other': [0.45454545454545453, 0.5454545454545454],
'EnvironmentalScience PerformingArts': [0.5, 0.5], 'CommunityService ParentInvolvement':
[0.45454545454545453, 0.5454545454545454], 'CommunityService Literacy': [0.45454545454545453,
0.5454545454545454], 'Music TeamSports': [0.45454545454545453, 0.5454545454545454], 'ESL
Health_Wellness': [0.45454545454545453, 0.5454545454545454], 'History_Geography Other': [0.5,
0.5], 'EnvironmentalScience Other': [0.5, 0.5], 'FinancialLiteracy Health_LifeScience':
[0.47619047619047616, 0.5238095238095238], 'CommunityService Economics': [0.5238095238095238,
0.47619047619047616], 'AppliedSciences Warmth Care_Hunger': [0.47619047619047616,
0.5238095238095238], 'ParentInvolvement PerformingArts': [0.47619047619047616,
0.5238095238095238], 'CommunityService NutritionEducation': [0.5238095238095238,
0.47619047619047616], 'FinancialLiteracy ForeignLanguages': [0.47619047619047616,
0.5238095238095238], 'Health_LifeScience ParentInvolvement': [0.47619047619047616,
0.5238095238095238], 'ForeignLanguages Other': [0.47619047619047616, 0.5238095238095238],
'CommunityService Health_LifeScience': [0.47619047619047616, 0.5238095238095238],
'ForeignLanguages Health_LifeScience': [0.47619047619047616, 0.5238095238095238],
'CommunityService Gym_Fitness': [0.47619047619047616, 0.5238095238095238], 'Music Other':
[0.5238095238095238, 0.47619047619047616], 'Economics Literature_Writing': [0.47619047619047616, 0
.5238095238095238], 'Civics_Government PerformingArts': [0.5238095238095238, 0.47619047619047616],
'Economics EnvironmentalScience': [0.47619047619047616, 0.5238095238095238], 'EarlyDevelopment Fin
ancialLiteracy': [0.47619047619047616, 0.5238095238095238], 'Mathematics NutritionEducation':
[0.47619047619047616, 0.5238095238095238], 'CharacterEducation Warmth Care_Hunger':
[0.5238095238095238, 0.47619047619047616], 'ForeignLanguages Health_Wellness':
[0.47619047619047616, 0.5238095238095238], 'Literacy Warmth Care_Hunger': [0.47619047619047616, 0.
5238095238095238], 'College_CareerPrep History_Geography': [0.5238095238095238,
0.47619047619047616], 'EnvironmentalScience Warmth Care_Hunger': [0.5238095238095238,
0.47619047619047616], 'CharacterEducation History_Geography': [0.47619047619047616,

0.4761904761904762], 'CharacterEducation History_Geography': [0.4761904761904762,
0.5238095238095238], 'Civics_Government NutritionEducation': [0.47619047619047616,
0.5238095238095238], 'Economics Literacy': [0.47619047619047616, 0.5238095238095238],
'PerformingArts SocialSciences': [0.5238095238095238, 0.47619047619047616], 'ESL
ParentInvolvement': [0.47619047619047616, 0.5238095238095238], 'CommunityService PerformingArts':
[0.47619047619047616, 0.5238095238095238], 'FinancialLiteracy Health_Wellness':
[0.47619047619047616, 0.5238095238095238], 'FinancialLiteracy History_Geography':
[0.47619047619047616, 0.5238095238095238], 'Literacy TeamSports': [0.47619047619047616,
0.5238095238095238], 'College_CareerPrep Gym_Fitness': [0.5238095238095238, 0.47619047619047616],
'Health_Wellness ParentInvolvement': [0.5238095238095238, 0.47619047619047616], 'Gym_Fitness
History_Geography': [0.47619047619047616, 0.5238095238095238], 'CommunityService SocialSciences':
[0.5238095238095238, 0.47619047619047616], 'Health_LifeScience Warmth Care_Hunger':
[0.47619047619047616, 0.5238095238095238], 'Extracurricular ParentInvolvement':
[0.47619047619047616, 0.5238095238095238], 'College_CareerPrep Warmth Care_Hunger':
[0.47619047619047616, 0.5238095238095238], 'CharacterEducation NutritionEducation':
[0.47619047619047616, 0.5238095238095238], 'ESL FinancialLiteracy': [0.47619047619047616,
0.5238095238095238], 'EnvironmentalScience Gym_Fitness': [0.47619047619047616,
0.5238095238095238], 'Gym_Fitness Literature_Writing': [0.5238095238095238, 0.47619047619047616],
'ESL SocialSciences': [0.47619047619047616, 0.5238095238095238], 'FinancialLiteracy
Literature_Writing': [0.47619047619047616, 0.5238095238095238], 'AppliedSciences
ForeignLanguages': [0.47619047619047616, 0.5238095238095238], 'NutritionEducation VisualArts': [0.
5238095238095238, 0.47619047619047616], 'Extracurricular SpecialNeeds': [0.47619047619047616, 0.52
38095238095238], 'AppliedSciences Civics_Government': [0.47619047619047616, 0.5238095238095238], '
College_CareerPrep Economics': [0.47619047619047616, 0.5238095238095238], 'Economics
NutritionEducation': [0.47619047619047616, 0.5238095238095238], 'Extracurricular Gym_Fitness': [0.
47619047619047616, 0.5238095238095238], 'Literacy NutritionEducation': [0.5238095238095238,
0.47619047619047616], 'EarlyDevelopment NutritionEducation': [0.47619047619047616,
0.5238095238095238], 'EarlyDevelopment TeamSports': [0.47619047619047616, 0.5238095238095238], 'He
alth_LifeScience TeamSports': [0.47619047619047616, 0.5238095238095238], 'History_Geography
ParentInvolvement': [0.47619047619047616, 0.5238095238095238], 'CharacterEducation Economics': [0.
5238095238095238, 0.47619047619047616], 'TeamSports VisualArts': [0.5238095238095238,
0.47619047619047616]}

2.4 Vectorizing Text data

2.4.1 Bag of words

In [38]:

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

print("="*100)


from sklearn.feature_extraction.text import CountVectorizer
vectorizer_essay = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer_essay.fit(X_train['essay'].values) # fit has to happen only on train data

featurenames=vectorizer_essay.get_feature_names()
# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer_essay.transform(X_train['essay'].values)
X_cv_essay_bow = vectorizer_essay.transform(X_cv['essay'].values)
X_test_essay_bow = vectorizer_essay.transform(X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)
```

```
(22445, 21) (22445,)
(11055, 21) (11055,)
(16500, 21) (16500,)
====================================================================================================

After vectorizations
(22445, 5000) (22445,)
(11055, 5000) (11055,)
(16500, 5000) (16500,)
====================================================================================================
```

In [39]:

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

print("="*100)

from sklearn.feature_extraction.text import CountVectorizer
vectorizer_title = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer_title.fit(X_train['project_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_bow = vectorizer_title.transform(X_train['project_title'].values)
X_cv_title_bow = vectorizer_title.transform(X_cv['project_title'].values)
X_test_title_bow = vectorizer_title.transform(X_test['project_title'].values)

print("After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
print("="*100)
```

```
(22445, 21) (22445,)
(11055, 21) (11055,)
(16500, 21) (16500,)
====================================================================================================

After vectorizations
(22445, 2647) (22445,)
(11055, 2647) (11055,)
(16500, 2647) (16500,)
====================================================================================================
```

2.4.2 TFIDF vectorizer

In [40]:

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

print("="*100)


from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_essay = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer_essay.fit(X_train['essay'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
featurenames=vectorizer_essay.get_feature_names()
X_train_essay_tfidf = vectorizer_essay.transform(X_train['essay'].values)
X_cv_essay_tfidf = vectorizer_essay.transform(X_cv['essay'].values)
X_test_essay_tfidf = vectorizer_essay.transform(X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
print("="*100)
```

```
(22445, 21) (22445,)
(11055, 21) (11055,)
(16500, 21) (16500,)
====================================================================================================

After vectorizations
(22445, 5000) (22445,)
(11055, 5000) (11055,)
(16500, 5000) (16500,)
====================================================================================================
```

In [41]:

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

print("="*100)


from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_title = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer_title.fit(X_train['project_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_title_tfidf = vectorizer_title.transform(X_train['project_title'].values)
X_cv_title_tfidf = vectorizer_title.transform(X_cv['project_title'].values)
X_test_title_tfidf = vectorizer_title.transform(X_test['project_title'].values)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
print("="*100)
```

```
(22445, 21) (22445,)
(11055, 21) (11055,)
(16500, 21) (16500,)
====================================================================================================

After vectorizations
(22445, 2647) (22445,)
(11055, 2647) (11055,)
(16500, 2647) (16500,)
====================================================================================================
```

2.4.3 AVG W2V for Essays

AVG W2V for Essays

In [42]:

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

Using Train Data

In [43]:

```
# average Word2Vec
# compute average word2vec for each review.
from scipy.sparse import csr_matrix
avg_w2v_vectors_for_essays_tr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_for_essays_tr.append(vector)
```

```
print(len(avg_w2v_vectors_for_essays_tr))
print(len(avg_w2v_vectors_for_essays_tr[0]))

avg_w2v_vectors_for_essays_tr=csr_matrix(avg_w2v_vectors_for_essays_tr)
```

```
100%|████████████████████████████████████████████████████████████████| 22445/22445
[00:19<00:00, 1139.38it/s]
```

```
22445
300
```

Using Test Data

In [44]:

```
# average Word2Vec
# compute average word2vec for each review.
from scipy.sparse import csr_matrix
avg_w2v_vectors_for_essays_te = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_for_essays_te.append(vector)

print(len(avg_w2v_vectors_for_essays_te))
print(len(avg_w2v_vectors_for_essays_te[0]))

avg_w2v_vectors_for_essays_te=csr_matrix(avg_w2v_vectors_for_essays_te)
```

```
100%|████████████████████████████████████████████████████████████████| 16500/16500
[00:13<00:00, 1195.74it/s]
```

```
16500
300
```

Using CV Data

In [45]:

```
# average Word2Vec
# compute average word2vec for each review.
from scipy.sparse import csr_matrix
avg_w2v_vectors_for_essays_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_for_essays_cv.append(vector)

print(len(avg_w2v_vectors_for_essays_cv))
print(len(avg_w2v_vectors_for_essays_cv[0]))

avg_w2v_vectors_for_essays_cv=csr_matrix(avg_w2v_vectors_for_essays_cv)
```

```
100%|████████████████████████████████████████████████████████████████| 11055/11055
[00:09<00:00, 1164.27it/s]
```

```
11055
```

300

AVG W2V for Titles

Using Train Data

In [46]:

```
# average Word2Vec
# compute average word2vec for each review.
from scipy.sparse import csr_matrix
avg_w2v_vectors_for_titles_tr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_for_titles_tr.append(vector)

print(len(avg_w2v_vectors_for_titles_tr))
print(len(avg_w2v_vectors_for_titles_tr[0]))

avg_w2v_vectors_for_titles_tr=csr_matrix(avg_w2v_vectors_for_titles_tr)
```

```
100%|██████████████████████████████████████████████████████████████████████████| 22445/22445
[00:00<00:00, 81926.13it/s]
```

```
22445
300
```

Using Test Data

In [47]:

```
# average Word2Vec
# compute average word2vec for each review.
from scipy.sparse import csr_matrix
avg_w2v_vectors_for_titles_te = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_for_titles_te.append(vector)

print(len(avg_w2v_vectors_for_titles_te))
print(len(avg_w2v_vectors_for_titles_te[0]))

avg_w2v_vectors_for_titles_te=csr_matrix(avg_w2v_vectors_for_titles_te)
```

```
100%|██████████████████████████████████████████████████████████████████████████| 16500/16500
[00:00<00:00, 70596.70it/s]
```

```
16500
300
```

Using CV Data

In [48]:

In [48]:

```
# average Word2Vec
# compute average word2vec for each review.
from scipy.sparse import csr_matrix
avg_w2v_vectors_for_titles_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_for_titles_cv.append(vector)

print(len(avg_w2v_vectors_for_titles_cv))
print(len(avg_w2v_vectors_for_titles_cv[0]))

avg_w2v_vectors_for_titles_cv=csr_matrix(avg_w2v_vectors_for_titles_cv)
```

```
100%|██████████████████████████████████████████████████████████████████████████| 11055/11055
[00:00<00:00, 81712.40it/s]

11055
300
```

2.4.4 Using Pretrained Models: TFIDF weighted W2V

TFIDF weighted W2V for Essays

Using Train Data

In [49]:

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

In [50]:

```
# average Word2Vec
# compute average word2vec for each review.
from scipy.sparse import csr_matrix
tfidf_w2v_vectors_for_essays_tr = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(X_train['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_for_essays_tr.append(vector)

print(len(tfidf_w2v_vectors_for_essays_tr))
print(len(tfidf_w2v_vectors_for_essays_tr[0]))

tfidf_w2v_vectors_for_essays_tr=csr_matrix(tfidf_w2v_vectors_for_essays_tr)
```

```
100%|████████████████████████████████████████████████████████████████| 22445/22445 [03:
32<00:00, 105.54it/s]
```

```
22445
300
```

Using Test Data

In [51]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_for_essays_te = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(X_test['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_for_essays_te.append(vector)

print(len(tfidf_w2v_vectors_for_essays_te))
print(len(tfidf_w2v_vectors_for_essays_te[0]))

tfidf_w2v_vectors_for_essays_te=csr_matrix(tfidf_w2v_vectors_for_essays_te)
```

```
100%|████████████████████████████████████████████████████████████████| 16500/16500 [02:
36<00:00, 105.65it/s]
```

```
16500
300
```

Using CV Data

In [52]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_for_essays_cv = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(X_cv['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_for_essays_cv.append(vector)

print(len(tfidf_w2v_vectors_for_essays_cv))
print(len(tfidf_w2v_vectors_for_essays_cv[0]))

tfidf_w2v_vectors_for_essays_cv=csr_matrix(tfidf_w2v_vectors_for_essays_cv)
```

```
100%|████████████████████████████████████████████████████████████████████████████| 11055/11055 [01:
40<00:00, 110.12it/s]
```

```
11055
300
```

TFIDF weighted W2V for Titles

Using Train Data

In [53]:

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

In [54]:

```
# average Word2Vec
# compute average word2vec for each review.
from scipy.sparse import csr_matrix
tfidf_w2v_vectors_for_titles_tr = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(X_train['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_for_titles_tr.append(vector)

print(len(tfidf_w2v_vectors_for_titles_tr))
print(len(tfidf_w2v_vectors_for_titles_tr[0]))

tfidf_w2v_vectors_for_titles_tr=csr_matrix(tfidf_w2v_vectors_for_titles_tr)
```

```
100%|████████████████████████████████████████████████████████████████████████████| 22445/22445
[00:00<00:00, 48304.27it/s]
```

```
22445
300
```

Using Test Data

In [55]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_for_titles_te = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(X_test['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
```

```
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_for_titles_te.append(vector)

print(len(tfidf_w2v_vectors_for_titles_te))
print(len(tfidf_w2v_vectors_for_titles_te[0]))

tfidf_w2v_vectors_for_titles_te=csr_matrix(tfidf_w2v_vectors_for_titles_te)
```

```
100%|████████████████████████████████████████████████████████████████| 16500/16500
[00:00<00:00, 55398.54it/s]
```

```
16500
300
```

Using CV Data

In [56]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_for_titles_cv = []; # the avg-w2v for each sentence/review is stored in this lis
t
for sentence in tqdm(X_cv['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_for_titles_cv.append(vector)

print(len(tfidf_w2v_vectors_for_titles_cv))
print(len(tfidf_w2v_vectors_for_titles_cv[0]))

tfidf_w2v_vectors_for_titles_cv=csr_matrix(tfidf_w2v_vectors_for_titles_cv)
```

```
100%|████████████████████████████████████████████████████████████████| 11055/11055
[00:00<00:00, 59911.58it/s]
```

```
11055
300
```

2.5 Vectorizing Numerical features

In [57]:

```
X_train_price_norm = (X_train['price'].values.reshape(-1,1))
X_cv_price_norm = (X_cv['price'].values.reshape(-1,1))
X_test_price_norm = (X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
```

```
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
==========================================================================================
```

In [58]:

```
X_train_teacher_norm = (X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1
))
X_cv_teacher_norm = (X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_teacher_norm = (X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_teacher_norm.shape, y_train.shape)
print(X_cv_teacher_norm.shape, y_cv.shape)
print(X_test_teacher_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
==========================================================================================
```

In [59]:

```
X_train_quantity_norm = (X_train['quantity'].values.reshape(-1,1))
X_cv_quantity_norm = (X_cv['quantity'].values.reshape(-1,1))
X_test_quantity_norm = (X_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_cv_quantity_norm.shape, y_cv.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
==========================================================================================
```

2.6 Merging all the above features

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

In [60]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((X_train_essay_bow,X_train_title_bow, X_train_state_response_coding,
X_train_teacher_prefix_response_coding,
X_train_project_grade_category_response_coding,X_train_clean_categories_response_coding,X_train_cle
an_subcategories_response_coding,X_train_price_norm,X_train_teacher_norm,X_train_quantity_norm))
X_cr = hstack((X_cv_essay_bow,X_cv_title_bow, X_cv_state_response_coding,
X_cv_teacher_prefix_response_coding,
X_cv_project_grade_category_response_coding,X_cv_clean_categories_response_coding,X_cv_clean_subcat
egories_response_coding,X_cv_price_norm,X_cv_teacher_norm,X_cv_quantity_norm))
X_te = hstack((X_test_essay_bow, X_test_title_bow,X_test_state_response_coding,
X_test_teacher_prefix_response_coding,
X_test_project_grade_category_response_coding,X_test_clean_categories_response_coding,X_test_clean_
subcategories_response_coding,X_test_quantity_norm,X_test_teacher_norm,X_test_quantity_norm))

print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 7660) (22445,)
(11055, 7660) (11055,)
(16500, 7660) (16500,)
==================================================================================================
```

In [61]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X2_tr = hstack((X_train_essay_tfidf,X_train_title_tfidf, X_train_state_response_coding, X_train_tea
cher_prefix_response_coding,
X_train_project_grade_category_response_coding,X_train_clean_categories_response_coding,X_train_cle
an_subcategories_response_coding,X_train_price_norm,X_train_teacher_norm,X_train_quantity_norm)).t
ocsr()
X2_cr = hstack((X_cv_essay_tfidf,X_cv_title_tfidf, X_cv_state_response_coding,
X_cv_teacher_prefix_response_coding,
X_cv_project_grade_category_response_coding,X_cv_clean_categories_response_coding,X_cv_clean_subcat
egories_response_coding,X_cv_price_norm,X_cv_teacher_norm,X_cv_quantity_norm)).tocsr()
X2_te = hstack((X_test_essay_tfidf, X_test_title_tfidf,X_test_state_response_coding,
X_test_teacher_prefix_response_coding,
X_test_project_grade_category_response_coding,X_test_clean_categories_response_coding,X_test_clean_
subcategories_response_coding,X_test_quantity_norm,X_test_teacher_norm,X_test_quantity_norm)).tocs
r()

print("Final Data matrix")
print(X2_tr.shape, y_train.shape)
print(X2_cr.shape, y_cv.shape)
print(X2_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 7660) (22445,)
(11055, 7660) (11055,)
(16500, 7660) (16500,)
==================================================================================================
```

In [62]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X3_tr = hstack((avg_w2v_vectors_for_essays_tr,avg_w2v_vectors_for_titles_tr,
X_train_state_response_coding, X_train_teacher_prefix_response_coding,
X_train_project_grade_category_response_coding,X_train_clean_categories_response_coding,X_train_cle
an_subcategories_response_coding,X_train_price_norm,X_train_teacher_norm,X_train_quantity_norm)).t
ocsr()
X3_cr = hstack((avg_w2v_vectors_for_essays_cv,avg_w2v_vectors_for_titles_cv,
X_cv_state_response_coding, X_cv_teacher_prefix_response_coding,
X_cv_project_grade_category_response_coding,X_cv_clean_categories_response_coding,X_cv_clean_subcat
egories_response_coding,X_cv_price_norm,X_cv_teacher_norm,X_cv_quantity_norm)).tocsr()
X3_te = hstack((avg_w2v_vectors_for_essays_te,
avg_w2v_vectors_for_titles_te,X_test_state_response_coding, X_test_teacher_prefix_response_coding,
X_test_project_grade_category_response_coding,X_test_clean_categories_response_coding,X_test_clean_
subcategories_response_coding,X_test_quantity_norm,X_test_teacher_norm,X_test_quantity_norm)).tocs
r()

print("Final Data matrix")
print(X3_tr.shape, y_train.shape)
print(X3_cr.shape, y_cv.shape)
print(X3_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 613) (22445,)
(11055, 613) (11055,)
(16500, 613) (16500,)
==================================================================================================
```

In [63]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X4_tr = hstack((tfidf_w2v_vectors_for_essays_tr,tfidf_w2v_vectors_for_titles_tr,
X_train_state_response_coding, X_train_teacher_prefix_response_coding,
X_train_project_grade_category_response_coding,X_train_clean_categories_response_coding,X_train_cle
an_subcategories_response_coding,X_train_price_norm,X_train_teacher_norm,X_train_quantity_norm)).t
ocsr()
X4_cr = hstack((tfidf_w2v_vectors_for_essays_cv,tfidf_w2v_vectors_for_titles_cv,
X_cv_state_response_coding, X_cv_teacher_prefix_response_coding,
X_cv_project_grade_category_response_coding,X_cv_clean_categories_response_coding,X_cv_clean_subcat
egories_response_coding,X_cv_price_norm,X_cv_teacher_norm,X_cv_quantity_norm)).tocsr()
X4_te = hstack((tfidf_w2v_vectors_for_essays_te,
tfidf_w2v_vectors_for_titles_te,X_test_state_response_coding,
X_test_teacher_prefix_response_coding,
X_test_project_grade_category_response_coding,X_test_clean_categories_response_coding,X_test_clean_
subcategories_response_coding,X_test_quantity_norm,X_test_teacher_norm,X_test_quantity_norm)).tocs
r()

print("Final Data matrix")
print(X4_tr.shape, y_train.shape)
print(X4_cr.shape, y_cv.shape)
print(X4_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 613) (22445,)
(11055, 613) (11055,)
(16500, 613) (16500,)
====================================================================================================
```

Assignment 9: RF and GBDT

2.7 Applying Random Forest

2.7.1 Applying Random Forests on BOW, SET 1

In [64]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,precision_score,recall_score

parameters = {'n_estimators': [5, 10, 50, 100, 200, 500, 1000],'max_depth':  [2, 3, 4, 5, 6, 7, 8, 9
, 10]}
rf = RandomForestClassifier(max_features='sqrt',class_weight="balanced")
clf = GridSearchCV(rf, parameters, scoring = 'f1_weighted', cv=3 )
clf.fit(X_tr, y_train)

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

In [65]:

```
print("Model with best parameters :\n",clf.best_estimator_)
print("Accuracy of the model : ",clf.score(X_te, y_test))
# Optimal value of number of base learners
optimal_learners = clf.best_estimator_.n_estimators
print("The optimal number of base learners is : ",optimal_learners)

optimal_depth=clf.best_estimator_.max_depth
print("The optimal number of depth is : ",optimal_depth)
```

```
Model with best parameters :
 RandomForestClassifier(bootstrap=True, class_weight='balanced',
```

```
            criterion='gini', max_depth=10, max_features='sqrt',
            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)
Accuracy of the model :  0.8067530431730241
The optimal number of base learners is :  500
The optimal number of depth is :  10
```

In [66]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=train_auc, name = 't
rain')
trace2 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=cv_auc, name = 'Cros
s validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='n_estimators'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [67]:

```
clf.best_params_
```

Out[67]:

```
{'max_depth': 10, 'n_estimators': 500}
```

The best value of max depth obtained from the plot is 10 and number of estimators is 500.

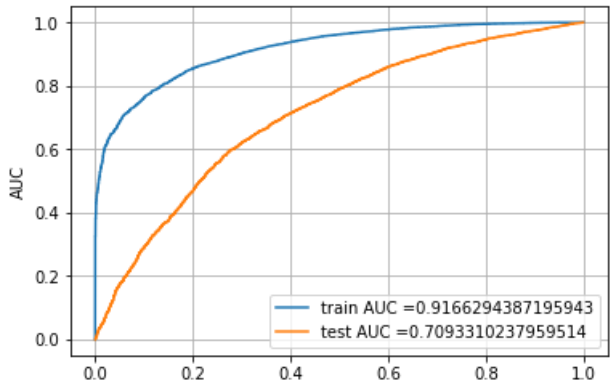
In [68]:

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

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

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

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

In [75]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier

clf = RandomForestClassifier(max_depth=10, n_estimators=500,class_weight="balanced",max_features='
sqrt')
clf.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X_tr)
preds = y_train_pred[:,1] #code copied from https://stackoverflow.com/questions/25009284/how-to-pl
ot-roc-curve-in-python to remove the error-bad input shape
y_test_pred = clf.predict_proba(X_te)
preds2=y_test_pred[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, preds)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, preds2)

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.ylabel("AUC")

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



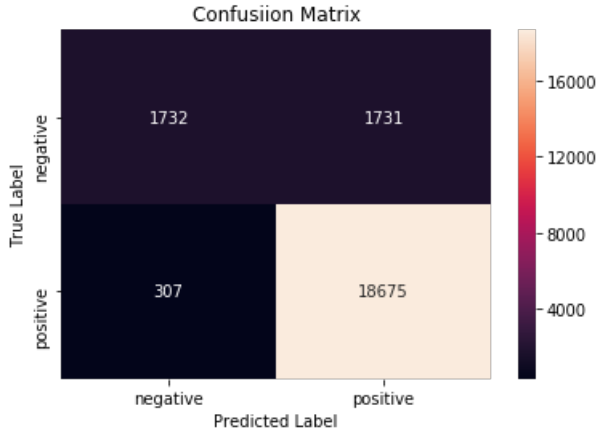
- Train AUC is 0.9166.
- Test AUC is 0.7093 represent the prediction level on the test dataset. In other words if a data point is provided the probabilty of classifying it correctly after the training has been done is 70.93 %.

In [77]:

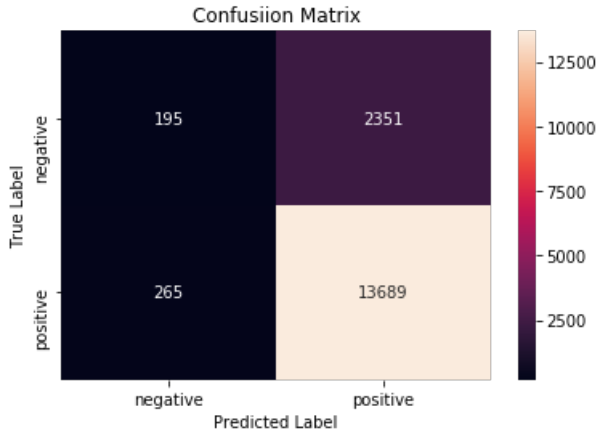
```
print("="*100)
from sklearn.metrics import confusion_matrix
import seaborn as sns
class_label = ["negative", "positive"]
```

```
# Reference: https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

print("Train confusion matrix")
cm=confusion_matrix(y_train, predict(preds, tr_thresholds, train_fpr, train_fpr))
df= pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Test confusion matrix")

cm=confusion_matrix(y_test, predict(preds2, tr_thresholds, test_fpr, test_fpr))
df = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.474
```




```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.481
```



2.7.2 Applying Random Forests on TFIDF, SET 2

In [69]:

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

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,precision_score,recall_score


parameters = {'n_estimators': [5, 10, 50, 100, 200, 500,1000],'max_depth': [2, 3, 4, 5, 6, 7, 8, 9,
10]}
rf = RandomForestClassifier(max_features='sqrt',class_weight="balanced")
clf = GridSearchCV(rf, parameters, scoring = 'f1_weighted', cv=3 )
clf.fit(X2_tr, y_train)

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

In [70]:

```
print("Model with best parameters :\n",clf.best_estimator_)
print("Accuracy of the model : ",clf.score(X2_te, y_test))
# Optimal value of number of base learners
optimal_learners = clf.best_estimator_.n_estimators
print("The optimal number of base learners is : ",optimal_learners)

optimal_depth=clf.best_estimator_.max_depth
print("The optimal number of depth is : ",optimal_depth)
```

```
Model with best parameters :
 RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=10, max_features='sqrt',
            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)
Accuracy of the model :  0.8052131041278981
The optimal number of base learners is :  1000
The optimal number of depth is :  10
```

In [71]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=train_auc, name = 't
rain')
trace2 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=cv_auc, name = 'Cros
s validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='n_estimators'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [72]:

```
clf.best_params_
```

Out[72]:

```
{'max_depth': 10, 'n_estimators': 1000}
```

The best value of max depth obtained from the plot is 10 and number of estimators is 1000.

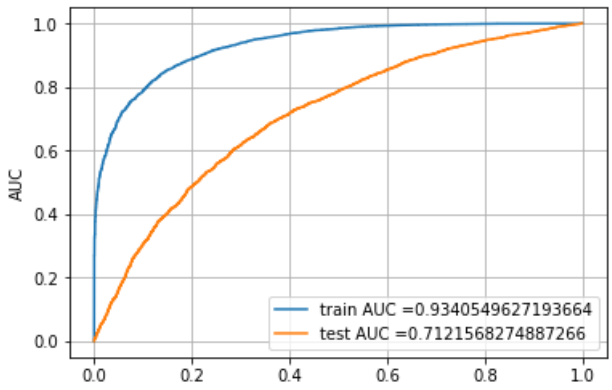
In [73]:

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

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

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

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

In [76]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier

clf = RandomForestClassifier(max_depth=10, n_estimators=1000,class_weight="balanced",max_features=
'sqrt')
clf.fit(X2_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X2_tr)
preds = y_train_pred[:,1] #code copied from https://stackoverflow.com/questions/25009284/how-to-pl
ot-roc-curve-in-python to remove the error-bad input shape
y_test_pred = clf.predict_proba(X2_te)
preds2=y_test_pred[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, preds)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, preds2)

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.ylabel("AUC")

plt.grid()
```

```
plt.show()
```

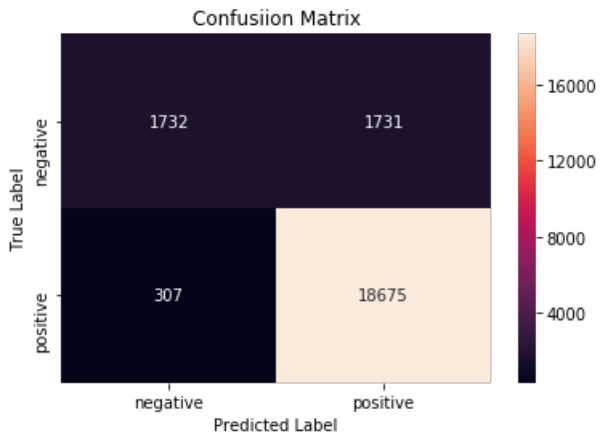


- Train AUC is 0.9340.
- Test AUC is 0.7121 represent the prediction level on the test dataset. In other words if a data point is provided the probabilty of classifying it correctly after the training has been done is 71.21 %.

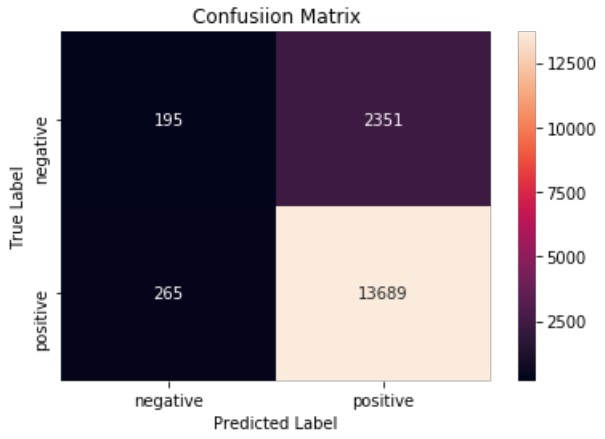
In [78]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
import seaborn as sns
class_label = ["negative", "positive"]

# Reference: https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

print("Train confusion matrix")
cm=confusion_matrix(y_train, predict(preds, tr_thresholds, train_fpr, train_fpr))
df= pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Test confusion matrix")

cm=confusion_matrix(y_test, predict(preds2, tr_thresholds, test_fpr, test_fpr))
df = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.474
```



```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.481
```



2.7.3 Applying Random Forests on AVG W2V, SET 3

In [79]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
import warnings
warnings.filterwarnings('always')
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,precision_score,recall_score

parameters = {'n_estimators': [5, 10, 50, 100, 200, 500],'max_depth': [2, 3, 4, 5, 6, 7, 8, 9, 10]}
rf = RandomForestClassifier(max_features='sqrt',class_weight="balanced")
clf = GridSearchCV(rf, parameters, cv=3 )
clf.fit(X3_tr, y_train)

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

```
C:\Users\dheer\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:125: FutureWarning:

You are accessing a training score ('mean_train_score'), which will not be available by default an
y more in 0.21. If you need training scores, please set return_train_score=True

C:\Users\dheer\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:125: FutureWarning:

You are accessing a training score ('std_train_score'), which will not be available by default any
more in 0.21. If you need training scores, please set return_train_score=True
```

In [80]:

```
print("Model with best parameters :\n",clf.best_estimator_)
print("Accuracy of the model : ",clf.score(X3_te, y_test))
# Optimal value of number of base learners
optimal_learners = clf.best_estimator_.n_estimators
print("The optimal number of base learners is : ",optimal_learners)

optimal_depth=clf.best_estimator_.max_depth
print("The optimal number of depth is : ",optimal_depth)
```

```
Model with best parameters :
 RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=10, max_features='sqrt',
            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)
Accuracy of the model :  0.8451515151515151
The optimal number of base learners is :  500
```

```
The optimal number of base learners is :  500
The optimal number of depth is :  10
```

In [81]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=train_auc, name = 't
rain')
trace2 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=cv_auc, name = 'Cros
s validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='n_estimators'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [82]:

```
clf.best_params_
```

Out[82]:

```
{'max_depth': 10, 'n_estimators': 500}
```

The best value of max depth obtained from the plot is 10 and number of estimators is 500.

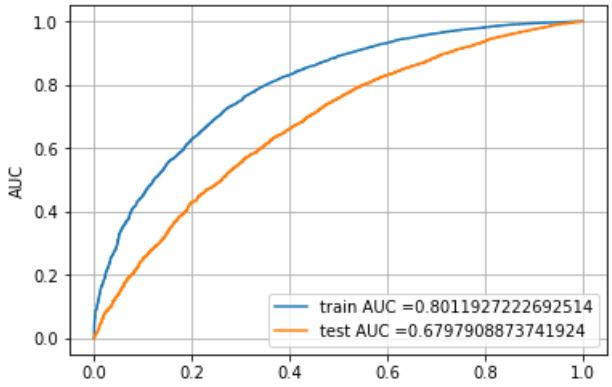
In [83]:

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

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

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

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

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

In [58]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier

clf = RandomForestClassifier(max_depth=10, n_estimators=500,class_weight="balanced",max_features='
sqrt')
clf.fit(X3_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X3_tr)
preds = y_train_pred[:,1] #code copied from https://stackoverflow.com/questions/25009284/how-to-pl
ot-roc-curve-in-python to remove the error-bad input shape
y_test_pred = clf.predict_proba(X3_te)
preds2=y_test_pred[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, preds)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, preds2)

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.ylabel("AUC")

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



- Train AUC is 0.8011.
- Test AUC is 0.6797 represent the prediction level on the test dataset. In other words if a data point is provided the probabilty of classifying it correctly after the training has been done is 67.97 %.

In [98]:

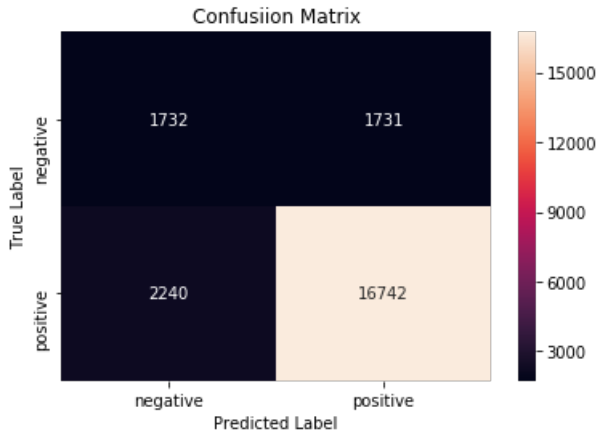
```
import warnings
warnings.filterwarnings('always')
print("="*100)
from sklearn.metrics import confusion_matrix
import seaborn as sns
class_label = ["negative", "positive"]

# Reference: https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

print("Train confusion matrix")
cm=confusion_matrix(y_train, predict(preds, tr_thresholds, train_fpr, train_fpr))
```

```
df= pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Test confusion matrix")

cm=confusion_matrix(y_test, predict(preds2, tr_thresholds, test_fpr, test_fpr))
df = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

==========

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.477
```

```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```




```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.491
```

```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```

2.7.4 Applying Random Forests on TFIDF W2V, SET 4

In [92]:

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

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridS earchCV
import warnings
warnings.filterwarnings('always')
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,precision_score,recall_score


parameters = {'n_estimators': [5, 10, 50, 100, 200, 500],'max_depth': [2, 3, 4, 5, 6, 7, 8, 9, 10]}
rf = RandomForestClassifier(max_features='sqrt',class_weight="balanced")
clf = GridSearchCV(rf, parameters, cv=3 , return_train_score=True)
clf.fit(X4_tr, y_train)

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

In [93]:

```
print("Model with best parameters :\n",clf.best_estimator_)
print("Accuracy of the model : ",clf.score(X4_te, y_test))
# Optimal value of number of base learners
optimal_learners = clf.best_estimator_.n_estimators
print("The optimal number of base learners is : ",optimal_learners)

optimal_depth=clf.best_estimator_.max_depth
print("The optimal number of depth is : ",optimal_depth)
```

```
Model with best parameters :
 RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=10, max_features='sqrt',
            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=200, n_jobs=None, oob_score=False,
            random_state=None, verbose=0, warm_start=False)
Accuracy of the model :  0.8382424242424242
The optimal number of base learners is :  200
The optimal number of depth is :  10
```

In [94]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=train_auc, name = 't
rain')
trace2 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=cv_auc, name = 'Cros
s validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='n_estimators'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

The best value of max depth obtained from the plot is 10 and number of estimators is 200.

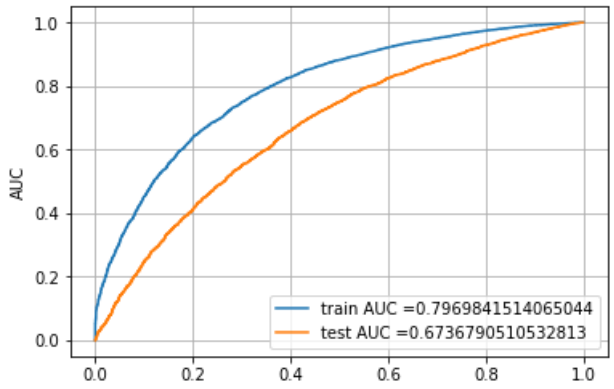
In [95]:

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

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

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

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

In [96]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier

clf = RandomForestClassifier(max_depth=10, n_estimators=200,class_weight="balanced",max_features='
sqrt')
clf.fit(X4_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X4_tr)
preds = y_train_pred[:,1] #code copied from https://stackoverflow.com/questions/25009284/how-to-pl
ot-roc-curve-in-python to remove the error-bad input shape
y_test_pred = clf.predict_proba(X4_te)
preds2=y_test_pred[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, preds)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, preds2)

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.ylabel("AUC")

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



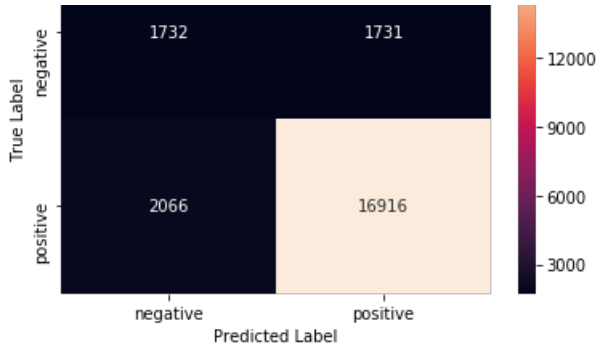
- Train AUC is 0.7969.
- Test AUC is 0.6736 represent the prediction level on the test dataset. In other words if a data point is provided the probabilty of classifying it correctly after the training has been done is 67.36 %.

In [61]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
import seaborn as sns
class_label = ["negative", "positive"]

# Reference: https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

print("Train confusion matrix")
cm=confusion_matrix(y_train, predict(preds, tr_thresholds, train_fpr, train_fpr))
df= pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Test confusion matrix")

cm=confusion_matrix(y_test, predict(preds2, tr_thresholds, test_fpr, test_fpr))
df = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

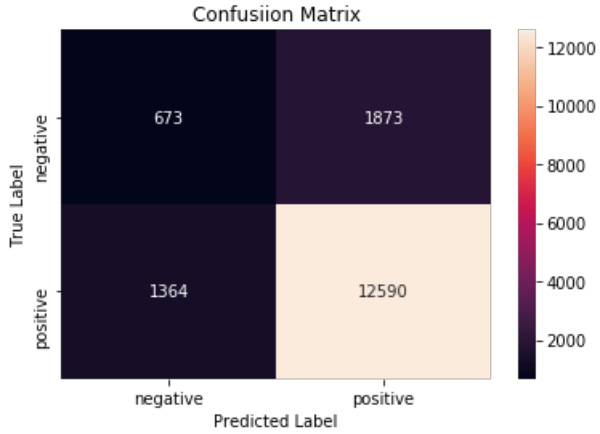
```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.479
```

```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```




```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.492
```

```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```



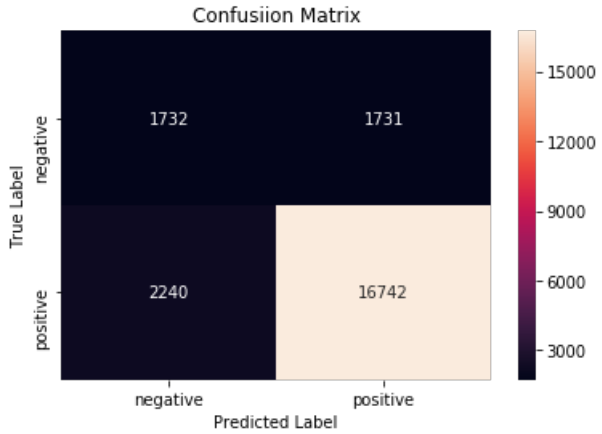
In [97]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
import seaborn as sns
class_label = ["negative", "positive"]

# Reference: https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

print("Train confusion matrix")
cm=confusion_matrix(y_train, predict(preds, tr_thresholds, train_fpr, train_fpr))
df= pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Test confusion matrix")

cm=confusion_matrix(y_test, predict(preds2, tr_thresholds, test_fpr, test_fpr))
df = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

==========

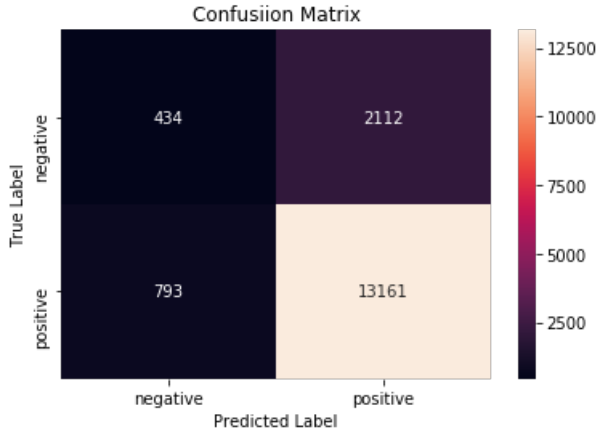
```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.477
```

```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```




```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.491
```

```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```



2.8 Applying GBDT

2.8.1 Applying XGBOOST on BOW, SET 1

In [76]:

```
# Importing library
from sklearn.ensemble import GradientBoostingClassifier

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

```
GBC = GradientBoostingClassifier(max_features='sqrt',subsample=0.1)
clf = GridSearchCV(GBC, param_grid, cv=3, n_jobs = -1,return_train_score=True)
clf.fit(X_tr, y_train)

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

In [110]:

```
print("Model with best parameters :\n",clf.best_estimator_)
print("Accuracy of the model : ",clf.score(X_te, y_test))

# Optimal value of number of base learners
optimal_learners = clf.best_estimator_.n_estimators
print("The optimal number of base learners is : ",optimal_learners)

# Optimal value of learning rate
optimal_depth = clf.best_estimator_.max_depth
print("\nThe optimal value of max depth is : ",optimal_depth)
```

```
Model with best parameters :
 GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=5,
              max_features='sqrt', 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=200,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=0.1, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
Accuracy of the model :  0.7812473804926823
The optimal number of base learners is :  200

The optimal value of max depth is :  5
```

In [111]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=train_auc, name = 't
rain')
trace2 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=cv_auc, name = 'Cros
s validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='n_estimators'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [112]:

```
clf.best_params_
```

Out[112]:

```
{'max_depth': 5, 'n_estimators': 200}
```

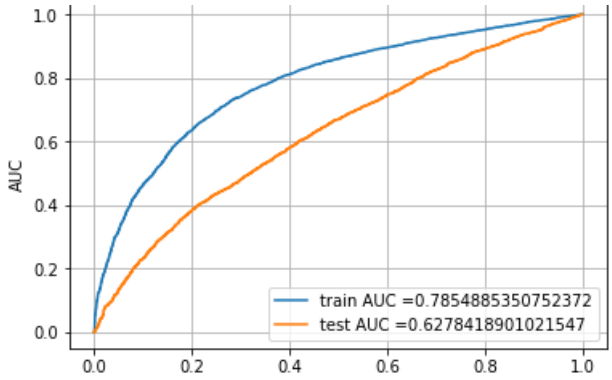
The best value of max depth obtained from the plot is 5 and number of estimators is 200.

In [113]:

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

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

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

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

In [114]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier

clf = GradientBoostingClassifier(max_features='sqrt',subsample=0.1,max_depth=5, n_estimators= 200)
clf.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X_tr)
preds = y_train_pred[:,1] #code copied from https://stackoverflow.com/questions/25009284/how-to-pl
ot-roc-curve-in-python to remove the error-bad input shape
y_test_pred = clf.predict_proba(X_te)
preds2=y_test_pred[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, preds)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, preds2)

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.ylabel("AUC")

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

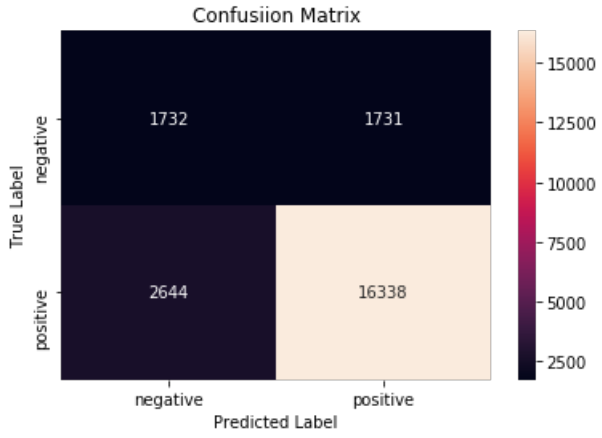


- Train AUC is 0.7854.
- Test AUC is 0.6278 represent the prediction level on the test dataset. In other words if a data point is provided the probabilty of classifying it correctly after the training has been done is 62.78 %.

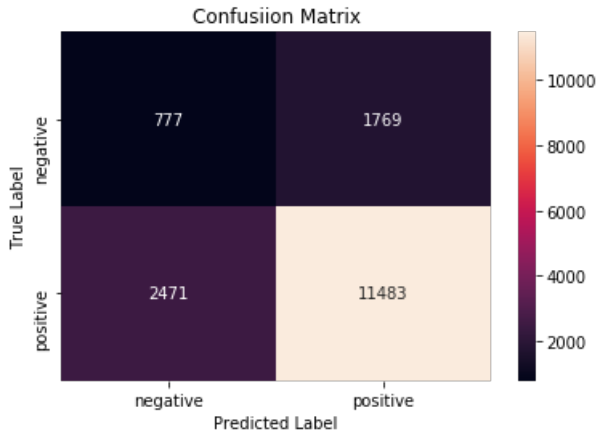
In [115]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
import seaborn as sns
class_label = ["negative", "positive"]

# Reference: https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

print("Train confusion matrix")
cm=confusion_matrix(y_train, predict(preds, tr_thresholds, train_fpr, train_fpr))
df= pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Test confusion matrix")

cm=confusion_matrix(y_test, predict(preds2, tr_thresholds, test_fpr, test_fpr))
df = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.775
```




```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999984572938835 for threshold 0.828
```



2.8.2 Applying XGBOOST on TFIDF, SET 2

In [77]:

```
# Importing library
from sklearn.ensemble import GradientBoostingClassifier

param_grid = {'n_estimators':[5, 10, 50, 100, 200, 500, 1000],  'max_depth': [1,5,10,50,100,150,200]
}
GBC = GradientBoostingClassifier(max_features='sqrt',subsample=0.1)
clf = GridSearchCV(GBC, param_grid, cv=3, n_jobs = -1,return_train_score=True)
clf.fit(X2_tr, y_train)

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

In [117]:

```
print("Model with best parameters :\n",clf.best_estimator_)
print("Accuracy of the model : ",clf.score(X2_te, y_test))

# Optimal value of number of base learners
optimal_learners = clf.best_estimator_.n_estimators
print("The optimal number of base learners is : ",optimal_learners)

# Optimal value of depth
optimal_depth = clf.best_estimator_.max_depth
print("\nThe optimal value of depth is : ",optimal_depth)
```

```
Model with best parameters :
 GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=5,
              max_features='sqrt', 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=150,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=0.1, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
Accuracy of the model :  0.7772341024469046
The optimal number of base learners is :  150

The optimal value of depth is :  5
```

In [119]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=train_auc, name = 't
rain')
trace2 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=cv_auc, name = 'Cros
s validation')
data = [trace1, trace2]
```

```
layout = go.Layout(scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='n_estimators'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [120]:

```
clf.best_params_
```

Out[120]:

```
{'max_depth': 5, 'n_estimators': 150}
```

The best value of max depth obtained from the plot is 5 and number of estimators is 150.

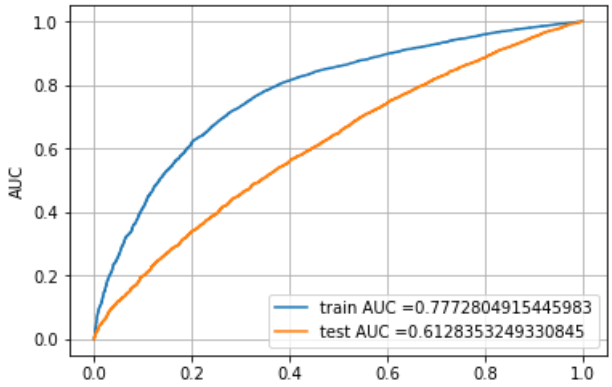
In [121]:

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

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

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

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

In [122]:

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

```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier

clf = GradientBoostingClassifier(max_features='sqrt',subsample=0.1,max_depth=5, n_estimators= 150)
clf.fit(X2_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X2_tr)
preds = y_train_pred[:,1] #code copied from https://stackoverflow.com/questions/25009284/how-to-pl
ot-roc-curve-in-python to remove the error-bad input shape
y_test_pred = clf.predict_proba(X2_te)
preds2=y_test_pred[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, preds)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, preds2)

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.ylabel("AUC")

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



- Train AUC is 0.7772.
- Test AUC is 0.6128 represent the prediction level on the test dataset. In other words if a data point is provided the probabilty of classifying it correctly after the training has been done is 61.28 %.

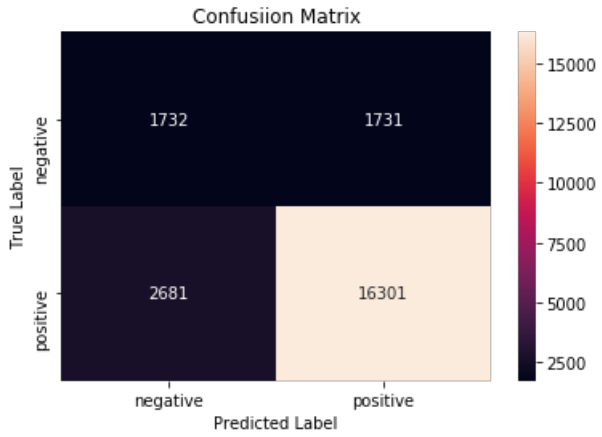
In [123]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
import seaborn as sns
class_label = ["negative", "positive"]

# Reference: https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

print("Train confusion matrix")
cm=confusion_matrix(y_train, predict(preds, tr_thresholds, train_fpr, train_fpr))
df= pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Test confusion matrix")

cm=confusion_matrix(y_test, predict(preds2, tr_thresholds, test_fpr, test_fpr))
df = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
```

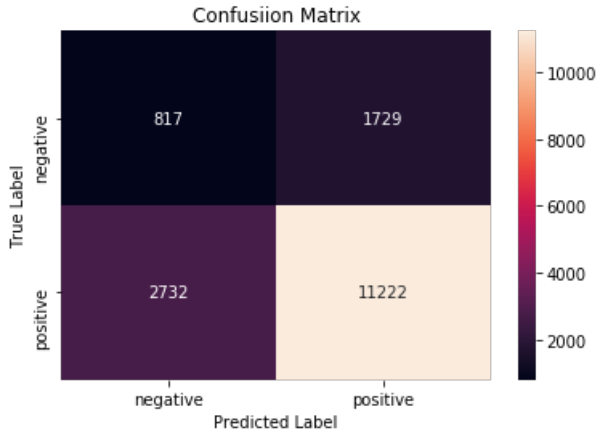
```
plt.ylabel("True Label")
plt.show()
```

==========

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.764
```




```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.82
```



2.8.3 Applying XGBOOST on AVG W2V, SET 3

In [78]:

```
# Importing library
from sklearn.ensemble import GradientBoostingClassifier

param_grid = {'n_estimators':[5, 10, 50, 100, 200, 500],  'max_depth': [2, 3, 4, 5, 6, 7, 8, 9, 10]}
GBC = GradientBoostingClassifier(max_features='sqrt',subsample=0.1)
clf = GridSearchCV(GBC, param_grid, cv=3, n_jobs = -1,return_train_score=True)
clf.fit(X3_tr, y_train)

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

In [62]:

```
print("Model with best parameters :\n",clf.best_estimator_)
print("Accuracy of the model : ",clf.score(X3_te, y_test))

# Optimal value of number of base learners
optimal_learners = clf.best_estimator_.n_estimators
print("The optimal number of base learners is : ",optimal_learners)
```

```
# Optimal value of depth
optimal_depth = clf.best_estimator_.max_depth
print("\nThe optimal value of depth is : ",optimal_depth)
```

```
Model with best parameters :
 GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=5,
              max_features='sqrt', 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=100,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=0.1, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
Accuracy of the model :  0.7852053291014787
The optimal number of base learners is :  100

The optimal value of depth is :  5
```

In [72]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=train_auc, name = 't
rain')
trace2 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=cv_auc, name = 'Cros
s validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='n_estimators'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [63]:

```
clf.best_params_
```

Out[63]:

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

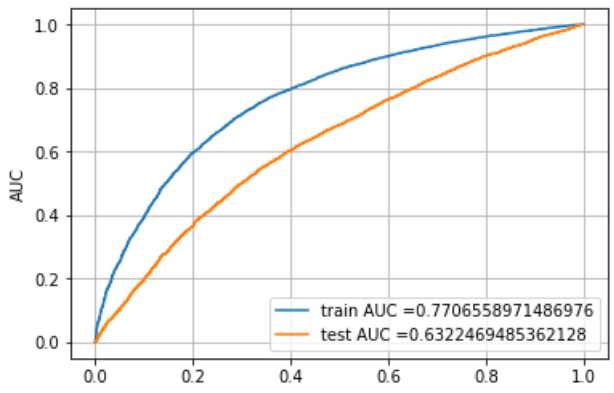
The best value of max depth obtained from the plot is 5 and number of estimators is 100.

In [64]:

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

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

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

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

In [65]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier

clf = GradientBoostingClassifier(max_features='sqrt',subsample=0.1,max_depth=5, n_estimators= 100)
clf.fit(X3_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
y_train_pred = clf.predict_proba(X3_tr)
preds = y_train_pred[:,1] #code copied from https://stackoverflow.com/questions/25009284/how-to-pl
ot-roc-curve-in-python to remove the error-bad input shape
y_test_pred = clf.predict_proba(X3_te)
preds2=y_test_pred[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, preds)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, preds2)

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.ylabel("AUC")

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



- Train AUC is 0.7706.
- Test AUC is 0.6322 represent the prediction level on the test dataset. In other words if a data point is provided the probabilty

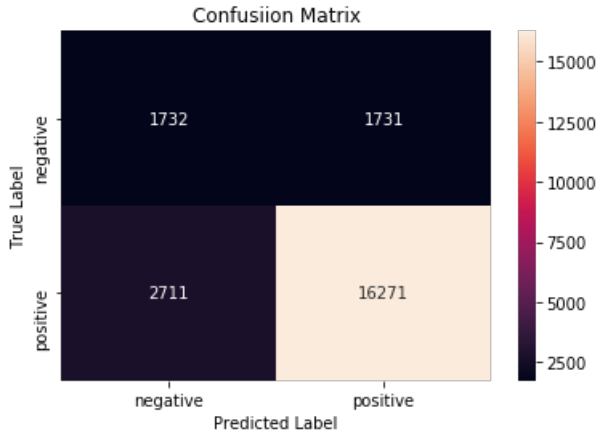
of classifying it correctly after the training has been done is 63.22 %.

In [66]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
import seaborn as sns
class_label = ["negative", "positive"]

# Reference: https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

print("Train confusion matrix")
cm=confusion_matrix(y_train, predict(preds, tr_thresholds, train_fpr, train_fpr))
df= pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Test confusion matrix")

cm=confusion_matrix(y_test, predict(preds2, tr_thresholds, test_fpr, test_fpr))
df = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

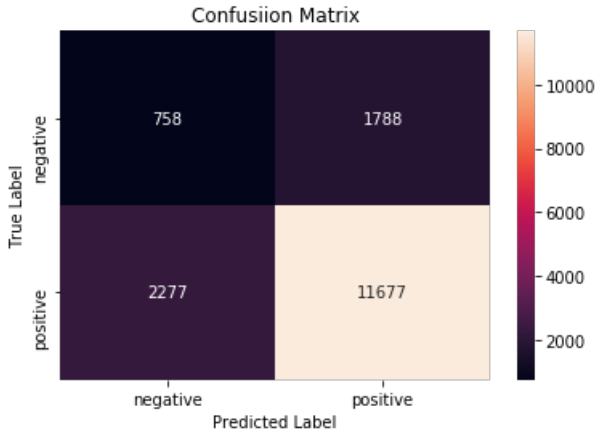
```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999997915341 for threshold 0.769
```

```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```




```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.827
```

```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```



2.8.4 Applying XGBOOST on TFIDF W2V, SET 4

In [79]:

```
# Importing library
from sklearn.ensemble import GradientBoostingClassifier

param_grid = {'n_estimators':[5, 10, 50, 100, 200, 500],  'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 10]}
GBC = GradientBoostingClassifier(max_features='sqrt',subsample=0.1)
clf = GridSearchCV(GBC, param_grid, cv=3, n_jobs = -1,return_train_score=True)
clf.fit(X4_tr, y_train)

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

In [64]:

```
print("Model with best parameters :\n",clf.best_estimator_)
print("Accuracy of the model : ",clf.score(X4_te, y_test))

# Optimal value of number of base learners
optimal_learners = clf.best_estimator_.n_estimators
print("The optimal number of base learners is : ",optimal_learners)

# Optimal value of depth
optimal_depth = clf.best_estimator_.max_depth
print("\nThe optimal value of depth is : ",optimal_depth)
```

```
Model with best parameters :
 GradientBoostingClassifier(criterion='friedman_mse', init=None,
              learning_rate=0.1, loss='deviance', max_depth=5,
              max_features='sqrt', 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=100,
              n_iter_no_change=None, presort='auto', random_state=None,
              subsample=0.1, tol=0.0001, validation_fraction=0.1,
              verbose=0, warm_start=False)
Accuracy of the model :  0.7811328432930681
The optimal number of base learners is :  100

The optimal value of depth is :  5
```

In [70]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=train_auc, name = 't
rain')
trace2 = go.Scatter3d(x=parameters['max_depth'],y=parameters['n_estimators'],z=cv_auc, name = 'Cros
s validation')
data = [trace1, trace2]
```

```
layout = go.Layout(scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='n_estimators'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

The best value of max depth obtained from the plot is 5 and number of estimators is 100.

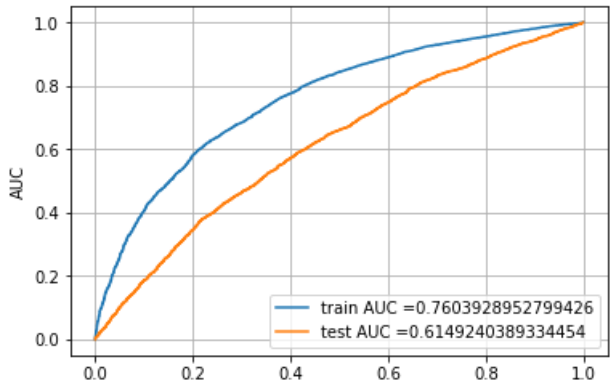
In [65]:

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

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

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

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

In [66]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt
from sklearn.metrics import roc_auc_score
from sklearn.tree import DecisionTreeClassifier

clf = GradientBoostingClassifier(max_features='sqrt',subsample=0.1,max_depth=5, n_estimators= 100)
clf.fit(X4_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs
```

```
y_train_pred = clf.predict_proba(X4_tr)
preds = y_train_pred[:,1] #code copied from https://stackoverflow.com/questions/25009284/how-to-plot-roc-curve-in-python to remove the error-bad input shape
y_test_pred = clf.predict_proba(X4_te)
preds2=y_test_pred[:,1]


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, preds)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, preds2)

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.ylabel("AUC")

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



- Train AUC is 0.7603.
- Test AUC is 0.6149 represent the prediction level on the test dataset. In other words if a data point is provided the probabilty of classifying it correctly after the training has been done is 61.49 %.

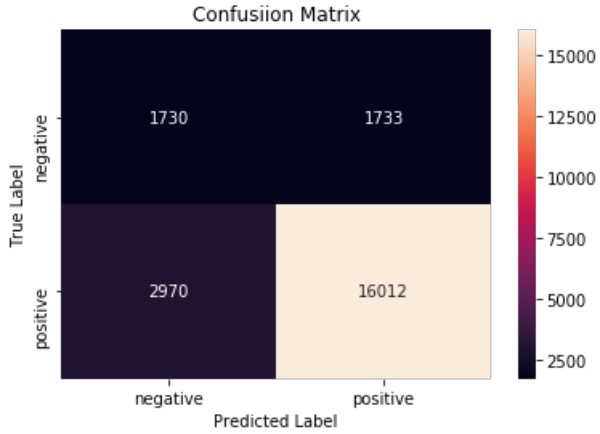
In [67]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
import seaborn as sns
class_label = ["negative", "positive"]

# Reference: https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

print("Train confusion matrix")
cm=confusion_matrix(y_train, predict(preds, tr_thresholds, train_fpr, train_fpr))
df= pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()

print("Test confusion matrix")

cm=confusion_matrix(y_test, predict(preds2, tr_thresholds, test_fpr, test_fpr))
df = pd.DataFrame(cm, index = class_label, columns = class_label)
sns.heatmap(df, annot = True, fmt = "d")
plt.title("Confusiion Matrix")
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.show()
```

```
====================================================================================================

Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999981238068975 for threshold 0.778
```

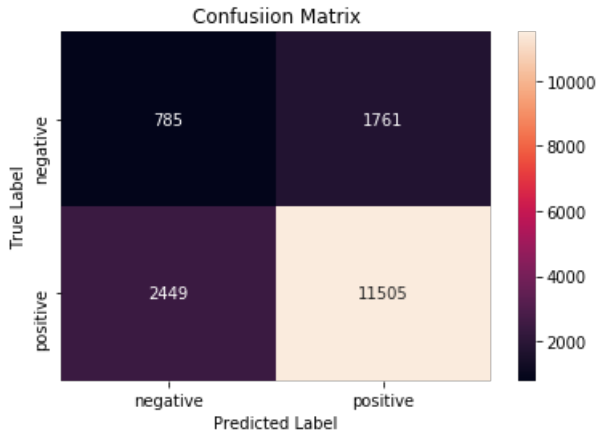
```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:
```

```
np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```




```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.826
```

```
C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead

C:\Users\dheer\Anaconda3\lib\site-packages\numpy\lib\type_check.py:546: DeprecationWarning:

np.asscalar(a) is deprecated since NumPy v1.16, use a.item() instead
```



3. Conclusion

In [99]:

```
# Please compare all your models using Prettytable library
# Reference: http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer",  "Max depth Hyper Parameter","No. of estimators Hyper Parameter","AU
C"]
x.add_row(["BOW", 10,500,0.7093])
x.add_row(["TFIDF",  10,1000,0.7121])
x.add_row(["AVG W2V",10,500,0.6797])
x.add_row(["TFIDF W2V",10,200, 0.6736])
print(" RANDOM FOREST ")
print(x)
```

```
x = PrettyTable()
x.field_names = ["Vectorizer",  "Max depth Hyper Parameter","No. of estimators Hyper Parameter","AU
C"]
x.add_row(["BOW",  5 ,200,0.6278])
x.add_row(["TFIDF",  5,150,0.6128])
x.add_row(["AVG W2V", 5,100,0.6322])
x.add_row(["TFIDF W2V",  5,100,0.6149])
print(" XGBOOST ")
print(x)
```

```
 RANDOM FOREST
+------------+---------------------------+-----------------------------------+--------+
| Vectorizer | Max depth Hyper Parameter | No. of estimators Hyper Parameter |  AUC   |
+------------+---------------------------+-----------------------------------+--------+
|    BOW     |             10            |                500                | 0.7093 |
|   TFIDF    |             10            |                1000               | 0.7121 |
|  AVG W2V   |             10            |                500                | 0.6797 |
| TFIDF W2V  |             10            |                200                | 0.6736 |
+------------+---------------------------+-----------------------------------+--------+
 XGBOOST
+------------+---------------------------+-----------------------------------+--------+
| Vectorizer | Max depth Hyper Parameter | No. of estimators Hyper Parameter |  AUC   |
+------------+---------------------------+-----------------------------------+--------+
|    BOW     |             5             |                200                | 0.6278 |
|   TFIDF    |             5             |                150                | 0.6128 |
|  AVG W2V   |             5             |                100                | 0.6322 |
| TFIDF W2V  |             5             |                100                | 0.6149 |
+------------+---------------------------+-----------------------------------+--------+
```

Random Forest

- In Random Forest the best value of Hyperparameter 'max_depth' 10 and 'n_estimators' lies between 200 -1000 according to the observations made with BOW,TFIDF,AVG W2V,TFIDF .
- Random Forest with BOW,TFIDF gives almost the same AUC i.e 71 % and AVG W2V,TFIDF W2V gives same AUC as 67% implies better performance of BOW and TFIDF as compared to AVG W2V and TFIDF W2V.
- Thus from the observations made, Random Forest is a 71%.

XGBoost

- In XGBoost the best value of Hyperparameter 'max_depth' is 5 and 'n_estimators' lies between 100 -200 according to the observations made with BOW,TFIDF,AVG W2V,TFIDF .
- Random Forest with BOW TFIDF, AVG W2V,TFIDF W2V gives almost similar AUC as 61%-63% implies similar performance of BOW ,TFIDF,AVG W2V and TFIDF W2V.
- Thus from the observations made, Random Forest is a suitable classifying model for predicting the project is approved or not in this case as it gives prediction rate between 60-65%.