Table of Contents

Author : SAURABH SINGHAI

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](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`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.

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

Import Libraries

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
```

Reading Data

In [2]:

```
# ******PLEASE NOTE--Considering 50K points as system becomes Unresponsive with higher no of point
s

project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
project_data= project_data.sample(n=50000, random_state=0)
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna(' ')
```

Approved And Non Approved Projects

In [3]:

```
y_value_counts = project_data['project_is_approved'].value_counts()

print("Project Not Approved & Approved Count=\n",y_value_counts)

print("Number of projects approved for funding ", y_value_counts[1], "=", (y_value_counts[1]/(y_val
ue_counts[1]+y_value_counts[0]))*100,"%")

print("Number of projects not approved for funding ", y_value_counts[0], "=", (y_value_counts[0]/(y
_value_counts[1]+y_value_counts[0]))*100,"%")
```

```
Project Not Approved & Approved Count=
 1    42460
0     7540
Name: project_is_approved, dtype: int64
Number of projects approved for funding  42460 = 84.92 %
Number of projects not approved for funding  7540 = 15.079999999999998 %
```

Project Dataframe shape and Column Values

In [4]:

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

```
Number of data points in project 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']
```

Resource data shape and Column Values

In [5]:

```
print("Number of data points in resource data", resource_data.shape)
print('*'*50)
print(resource_data.columns.values)
resource_data.head(5)
```

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

Out[5]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

Preprocessing of `project_subject_categories`

In [6]:

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

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

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

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

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

Preprocessing of `project_subject_subcategories`

In [7]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
```

```
# remove special characters from list of strings python:
https://stackoverflow.com/a/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]))
```

Preprocessing of `project_grade_category`

In [8]:

```
print(project_data["project_grade_category"].values[0:10])
```

```
['Grades 3-5' 'Grades 6-8' 'Grades 3-5' 'Grades 3-5' 'Grades PreK-2'
 'Grades PreK-2' 'Grades 3-5' 'Grades 3-5' 'Grades PreK-2' 'Grades 9-12']
```

In [9]:

```
project_data["project_grade_category"] =
project_data["project_grade_category"].str.replace("Grades ", "")
project_data["project_grade_category"] = project_data["project_grade_category"].str.replace("-", "_
")

print(project_data["project_grade_category"].values[0:10])
```

```
['3_5' '6_8' '3_5' '3_5' 'PreK_2' 'PreK_2' '3_5' '3_5' 'PreK_2' '9_12']
```

Create new column 'Essay' by merging all project Essays

In [10]:

```
# 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 [11]:

```
project_data['essay'].head(5)
```

Out[11]:

```
75155    Starting the new year off right sets the tone ...
77488    Have you ever worked so hard on a project only...
7803     My students come to class every day ready to l...
56268    \"We love science in your class!\" CJ exclaime...
46902    My students are caring, outgoing, and creative...
Name: essay, dtype: object
```

In [12]:

```
# printing some random reviews
print(project_data['essay'].values[0])
```

```
Starting the new year off right sets the tone for months to come. My class will be thrilled to rec
eive basic supplies to help them be successful.\r\n\r\nMy students are curious, inquisitive, and e
nthusiastic learners who enjoy school.\r\n\r\nOur school is a public community school in New York
City that receives Title I funding, which means that many students are eligible for free or reduce
d price lunch. Most of my students are English language learners. Our self-contained class is comp
rised of students with disabilities in second and third grade.We need printer ink so we can showca
se our wonderful work, and other supplies such as pocket charts for subject-specific word walls.\r
\n\r\nThe poetry book will align with our specialized phonics and reading program, and the Recipro
cal Teaching Strategies book will help us get where we need to be.\r\n\r\nChart paper is a staple
for any literacy or math lesson, and folders will help keep us organized. Ziplock pouches will att
ach to students' homework folders, making it simple and easy to transport school books home and ba
ck. \r\n\r\nPlease help us meet our needs with your support and generous donations. Thank
you!nannan
```

Use Decontraction function to decontract project essay

In [13]:

```
# 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 [14]:

```
sent = decontracted(project_data['essay'].values[1])
print(sent)
print("="*50)
```

```
Have you ever worked so hard on a project only to get it back from a teacher with a dismal grade?
Or have a loved one that tries so very hard in school but just does not seem to grasp the
concepts? That is how my students with special needs feel everyday! I create a classroom where eve
ryone succeeds.\r\nMy students all have mild to moderate disabilities.  The disabilities range fro
m various levels of autism, moderate learning disabilities, challenges with attention, to being
classified as intellectually impaired.  \r\nThese students are just wonderful people but face dail
y challenges that you and I could never fathom. Most of my students come from low socioeconomic ho
mes. Suffering from disabilities makes it difficult for them to read, comprehend, write, and solve
math equations using typical learning styles. Technology is a way to bridge that learning gap thes
e students struggle with each and every day.\r\nThese Chromebooks will be used in my classroom to
help students complete their Common Core assignments in all subject areas. Students will be able t
o use this technology to help with the 21st century skills needed to be successful with the new Co
mmon Core State Standards and daily life.\r\nThis technology will make a huge impact on their
lives.  We currently have a teacher computer, document camera, projector, printer, and one
```

```
chromebook per two students. The school itself has a few computer labs that it shares with the ent
ire student body. These Chromebooks will be a huge benefit to my students' lives.nannan
==================================================
```

Remove line breaks

In [15]:

```python
# \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)
```

```
Have you ever worked so hard on a project only to get it back from a teacher with a dismal grade?
Or have a loved one that tries so very hard in school but just does not seem to grasp the
concepts? That is how my students with special needs feel everyday! I create a classroom where eve
ryone succeeds.  My students all have mild to moderate disabilities.  The disabilities range from
various levels of autism, moderate learning disabilities, challenges with attention, to being
classified as intellectually impaired.    These students are just wonderful people but face daily
challenges that you and I could never fathom. Most of my students come from low socioeconomic home
s. Suffering from disabilities makes it difficult for them to read, comprehend, write, and solve m
ath equations using typical learning styles. Technology is a way to bridge that learning gap these
students struggle with each and every day.  These Chromebooks will be used in my classroom to help
students complete their Common Core assignments in all subject areas. Students will be able to use
this technology to help with the 21st century skills needed to be successful with the new Common C
ore State Standards and daily life.  This technology will make a huge impact on their lives.  We
currently have a teacher computer, document camera, projector, printer, and one chromebook per two
students. The school itself has a few computer labs that it shares with the entire student body. T
hese Chromebooks will be a huge benefit to my students' lives.nannan
```

Remove Special Chars

In [16]:

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

```
Have you ever worked so hard on a project only to get it back from a teacher with a dismal grade O
r have a loved one that tries so very hard in school but just does not seem to grasp the concepts
That is how my students with special needs feel everyday I create a classroom where everyone succe
eds My students all have mild to moderate disabilities The disabilities range from various levels
of autism moderate learning disabilities challenges with attention to being classified as
intellectually impaired These students are just wonderful people but face daily challenges that yo
u and I could never fathom Most of my students come from low socioeconomic homes Suffering from di
sabilities makes it difficult for them to read comprehend write and solve math equations using
typical learning styles Technology is a way to bridge that learning gap these students struggle wi
th each and every day These Chromebooks will be used in my classroom to help students complete the
ir Common Core assignments in all subject areas Students will be able to use this technology to he
lp with the 21st century skills needed to be successful with the new Common Core State Standards a
nd daily life This technology will make a huge impact on their lives We currently have a teacher c
omputer document camera projector printer and one chromebook per two students The school itself ha
s a few computer labs that it shares with the entire student body These Chromebooks will be a huge
benefit to my students lives nannan
```

Remove Stopwards and Join the essays

In [17]:

```python
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
```

```
            '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', "do
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 [18]:

```
# Combining all the above stundents
def Text_cleaner(data):
    from tqdm import tqdm
    preprocessed_essays = []
    # tqdm is for printing the status bar
    for sentance in tqdm(data.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.lower() not in stopwords)
        preprocessed_essays.append(sent.lower().strip())
    return preprocessed_essays
```

In [19]:

```
# after preprocesing
preprocessed_essays=Text_cleaner(project_data['essay'])
```

```
100%|████████████████████████████████████████████████████████████| 50000/50000
[00:32<00:00, 1543.31it/s]
```

In [20]:

```
preprocessed_essays[1]
```

Out[20]:

```
'ever worked hard project get back teacher dismal grade loved one tries hard school not seem grasp
concepts students special needs feel everyday create classroom everyone succeeds students mild mod
erate disabilities disabilities range various levels autism moderate learning disabilities
challenges attention classified intellectually impaired students wonderful people face daily chall
enges could never fathom students come low socioeconomic homes suffering disabilities makes
difficult read comprehend write solve math equations using typical learning styles technology way
bridge learning gap students struggle every day chromebooks used classroom help students complete
common core assignments subject areas students able use technology help 21st century skills needed
successful new common core state standards daily life technology make huge impact lives currently
teacher computer document camera projector printer one chromebook per two students school computer
labs shares entire student body chromebooks huge benefit students lives nannan'
```

Drop essay columns 1, 2, 3, 4

In [21]:

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

Out[21]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade
75155	144107	p064182	7414165942b20a8d7fe5bcdc96244624	Ms.	NY	2017-02-05 17:49:01	
77488	89277	p187708	5b42a9aa00917ac1716d8063aebc6318	Mrs.	CA	2016-05-27 14:44:25	
7803	123550	p142214	bec515840d4fb7d2ba1071211ba32231	Mrs.	CA	2016-12-09 19:23:16	
56268	104617	p098697	8131749e34b7ef3fa0890b5d840deb2a	Ms.	NC	2016-06-20 13:27:03	
46902	154452	p252651	d240517694ebcbe54a5ffa806a5ada2e	Ms.	MO	2016-11-09 15:54:10	

Preprocessing of `project_title`

In [22]:

```
# similarly you can preprocess the titles also
preprocessed_project_title=Text_cleaner(project_data['project_title'])
```

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:01<00:00, 30480.85it/s]
```

In [23]:

```
preprocessed_project_title[1]
```

Out[23]:

```
'keep spirit alive'
```

Drop column project_title and use Cleaned_Title

In [24]:

```
project_data['Cleaned_title']= preprocessed_project_title
project_data.drop(['project_title'], axis=1, inplace=True)
```

Add up the price based on project id

In [25]:

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

In [26]:

```
project_data.columns
```

Out[26]:

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

In [27]:

```
project_data.drop(['project_resource_summary'], axis=1, inplace=True)
project_data.drop(['Unnamed: 0'], axis=1, inplace=True)
project_data.drop(['id'], axis=1, inplace=True)
project_data.drop(['teacher_id'], axis=1, inplace=True)
```

Adding the word count of essay and title as new columns

In [28]:

```
project_data['essay_count']=project_data['essay'].str.len()
project_data['title_count']=project_data['Cleaned_title'].str.len()
```

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

-Length of words in essay
-Length of words in title
```

Splitting data into Test,Train,CV

In [29]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

from sklearn.model_selection import train_test_split

X_train,X_test,y_train, y_test=train_test_split(project_data, project_data['project_is_approved'],
test_size=0.3, stratify = project_data['project_is_approved'])
```

```
# intentionally taking less data in cv ; will not use it
X_train,X_cv,y_train,y_cv=train_test_split(X_train, y_train, test_size=0.3,
random_state=0,stratify=y_train)

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

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

```
(24500, 13)
(15000, 13)
(10500, 13)
(24500,)
(15000,)
(10500,)
```

Response coding for Categorical Data

Create function for Response Encoding

In [30]:

```
#https://stackoverflow.com/questions/11869910/pandas-filter-rows-of-dataframe-with-operator-chaini
ng
def mask(df, key, value):
    return df[df[key] == value]

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

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

    return prob_neg,prob_pos
```

Vectorize the Categorical Features - categories

In [31]:

```
#Clean Categories
cat_0_train = get_response(X_train['clean_categories'],y_train)[0]
cat_1_train = get_response(X_train['clean_categories'],y_train)[1]

cat_0_test = get_response(X_test['clean_categories'],y_test)[0]
cat_1_test = get_response(X_test['clean_categories'],y_test)[1]

cat_0_cv = get_response(X_cv['clean_categories'],y_cv)[0]
cat_1_cv = get_response(X_cv['clean_categories'],y_cv)[1]

print(cat_0_train)
print(cat_1_train)
```

```python
print(cat_0_test)
print(cat_1_test)

print(cat_0_cv)
print(cat_1_cv)
#========================================add to train df
cat_neg_train = []
cat_pos_train = []
for i in X_train['clean_categories']:
    cat_neg_train.append(cat_0_train[i])
    cat_pos_train.append(cat_1_train[i])
X_train['cat_0'] = cat_neg_train
X_train['cat_1'] = cat_pos_train

#========================================add to test df
cat_neg_test = []
cat_pos_test = []
for i in X_test['clean_categories']:
    cat_neg_test.append(cat_0_test[i])
    cat_pos_test.append(cat_1_test[i])
X_test['cat_0'] = cat_neg_test
X_test['cat_1'] = cat_pos_test

#========================================add to cv df
cat_neg_cv = []
cat_pos_cv = []
for i in X_cv['clean_categories']:
    cat_neg_cv.append(cat_0_cv[i])
    cat_pos_cv.append(cat_1_cv[i])

X_cv['cat_0'] = cat_neg_cv
X_cv['cat_1'] = cat_pos_cv
```

```
{'AppliedLearning': 0.19441210710128057, 'AppliedLearning Health_Sports': 0.19736842105263158,
'AppliedLearning History_Civics': 0.05, 'AppliedLearning Literacy_Language': 0.1282608695652174, '
AppliedLearning Math_Science': 0.16810344827586207, 'AppliedLearning Music_Arts':
0.28289473684210525, 'AppliedLearning SpecialNeeds': 0.17834394904458598, 'AppliedLearning Warmth
Care_Hunger': 0.5, 'Health_Sports': 0.15803336259877085, 'Health_Sports AppliedLearning':
0.13636363636363635, 'Health_Sports History_Civics': 0.2727272727272727, 'Health_Sports
Literacy_Language': 0.21212121212121213, 'Health_Sports Math_Science': 0.11864406779661017,
'Health_Sports Music_Arts': 0.19230769230769232, 'Health_Sports SpecialNeeds':
0.12380952380952381, 'Health_Sports Warmth Care_Hunger': 0.0, 'History_Civics':
0.15328467153284672, 'History_Civics AppliedLearning': 0.14285714285714285, 'History_Civics
Health_Sports': 0.5, 'History_Civics Literacy_Language': 0.09451219512195122, 'History_Civics
Math_Science': 0.10843373493975904, 'History_Civics Music_Arts': 0.07792207792207792,
'History_Civics SpecialNeeds': 0.1896551724137931, 'Literacy_Language': 0.12863385901577049,
'Literacy_Language AppliedLearning': 0.14864864864864866, 'Literacy_Language Health_Sports':
0.2857142857142857, 'Literacy_Language History_Civics': 0.12, 'Literacy_Language Math_Science':
0.12530048076923078, 'Literacy_Language Music_Arts': 0.15718157181571815, 'Literacy_Language
SpecialNeeds': 0.16554809843400448, 'Literacy_Language Warmth Care_Hunger': 0.0, 'Math_Science':
0.18515583729529847, 'Math_Science AppliedLearning': 0.17567567567567569, 'Math_Science
Health_Sports': 0.23529411764705882, 'Math_Science History_Civics': 0.16666666666666666,
'Math_Science Literacy_Language': 0.13285457809694792, 'Math_Science Music_Arts':
0.14360313315926893, 'Math_Science SpecialNeeds': 0.17551963048498845, 'Math_Science Warmth
Care_Hunger': 1.0, 'Music_Arts': 0.13784246575342465, 'Music_Arts AppliedLearning': 0.0,
'Music_Arts Health_Sports': 0.0, 'Music_Arts History_Civics': 0.3333333333333333, 'Music_Arts Spec
ialNeeds': 0.18181818181818182, 'Music_Arts Warmth Care_Hunger': 0.5, 'SpecialNeeds':
0.16967126193001061, 'SpecialNeeds Health_Sports': 0.2222222222222222, 'SpecialNeeds Music_Arts':
0.21311475409836064, 'SpecialNeeds Warmth Care_Hunger': 0.3333333333333333, 'Warmth Care_Hunger':
0.06643356643356643}
{'AppliedLearning': 0.8055878928987195, 'AppliedLearning Health_Sports': 0.8026315789473685,
'AppliedLearning History_Civics': 0.95, 'AppliedLearning Literacy_Language': 0.8717391304347826, '
AppliedLearning Math_Science': 0.8318965517241379, 'AppliedLearning Music_Arts':
0.7171052631578947, 'AppliedLearning SpecialNeeds': 0.821656050955414, 'AppliedLearning Warmth Car
e_Hunger': 0.5, 'Health_Sports': 0.8419666374012291, 'Health_Sports AppliedLearning':
0.8636363636363636, 'Health_Sports History_Civics': 0.7272727272727273, 'Health_Sports
Literacy_Language': 0.7878787878787878, 'Health_Sports Math_Science': 0.8813559322033898,
'Health_Sports Music_Arts': 0.8076923076923077, 'Health_Sports SpecialNeeds': 0.8761904761904762,
'Health_Sports Warmth Care_Hunger': 1.0, 'History_Civics': 0.8467153284671532, 'History_Civics
AppliedLearning': 0.8571428571428571, 'History_Civics Health_Sports': 0.5, 'History_Civics
Literacy_Language': 0.9054878048780488, 'History_Civics Math_Science': 0.891566265060241,
'History_Civics Music_Arts': 0.922077922077922, 'History_Civics SpecialNeeds': 0.8103448275862069,
'Literacy_Language': 0.8713661409842295, 'Literacy_Language AppliedLearning': 0.8513513513513513,
'Literacy_Language Health_Sports': 0.7142857142857143, 'Literacy_Language History_Civics': 0.88, '
Literacy_Language Math_Science': 0.8746995192307693, 'Literacy_Language Music_Arts':
```

0.8428184281842819, 'Literacy_Language SpecialNeeds': 0.8344519015659956, 'Literacy_Language
Warmth Care_Hunger': 1.0, 'Math_Science': 0.8148441627047015, 'Math_Science AppliedLearning':
0.8243243243243243, 'Math_Science Health_Sports': 0.7647058823529411, 'Math_Science
History_Civics': 0.8333333333333334, 'Math_Science Literacy_Language': 0.8671454219030521,
'Math_Science Music_Arts': 0.856396866840731, 'Math_Science SpecialNeeds': 0.8244803695150116,
'Math_Science Warmth Care_Hunger': 0.0, 'Music_Arts': 0.8621575342465754, 'Music_Arts
AppliedLearning': 1.0, 'Music_Arts Health_Sports': 1.0, 'Music_Arts History_Civics':
0.6666666666666666, 'Music_Arts SpecialNeeds': 0.8181818181818182, 'Music_Arts Warmth
Care_Hunger': 0.5, 'SpecialNeeds': 0.8303287380699894, 'SpecialNeeds Health_Sports':
0.7777777777777778, 'SpecialNeeds Music_Arts': 0.7868852459016393, 'SpecialNeeds Warmth
Care_Hunger': 0.6666666666666666, 'Warmth Care_Hunger': 0.9335664335664335}
{'AppliedLearning': 0.18860510805500982, 'AppliedLearning Health_Sports': 0.15476190476190477,
'AppliedLearning History_Civics': 0.125, 'AppliedLearning Literacy_Language': 0.1314878892733564,
'AppliedLearning Math_Science': 0.1643835616438356, 'AppliedLearning Music_Arts':
0.17894736842105263, 'AppliedLearning SpecialNeeds': 0.22727272727272727, 'AppliedLearning Warmth
Care_Hunger': 1.0, 'Health_Sports': 0.14265335235378032, 'Health_Sports AppliedLearning': 0.09375,
'Health_Sports History_Civics': 0.125, 'Health_Sports Literacy_Language': 0.09821428571428571, 'He
alth_Sports Math_Science': 0.17647058823529413, 'Health_Sports Music_Arts': 0.19230769230769232, '
Health_Sports SpecialNeeds': 0.10795454545454546, 'Health_Sports Warmth Care_Hunger': 0.0,
'History_Civics': 0.16872427983539096, 'History_Civics AppliedLearning': 0.5, 'History_Civics
Health_Sports': 0.0, 'History_Civics Literacy_Language': 0.08854166666666667, 'History_Civics
Math_Science': 0.17073170731707318, 'History_Civics Music_Arts': 0.24444444444444444,
'History_Civics SpecialNeeds': 0.24324324324324326, 'Literacy_Language': 0.13441024078024993,
'Literacy_Language AppliedLearning': 0.14634146341463414, 'Literacy_Language Health_Sports': 0.0,
'Literacy_Language History_Civics': 0.13861386138613863, 'Literacy_Language Math_Science':
0.12323943661971831, 'Literacy_Language Music_Arts': 0.1615720524017467, 'Literacy_Language
SpecialNeeds': 0.1410488245931284, 'Literacy_Language Warmth Care_Hunger': 0.0, 'Math_Science':
0.1896404109589041, 'Math_Science AppliedLearning': 0.19428571428571428, 'Math_Science
Health_Sports': 0.1724137931034483, 'Math_Science History_Civics': 0.15584415584415584,
'Math_Science Literacy_Language': 0.16614420062695925, 'Math_Science Music_Arts':
0.1902439024390244, 'Math_Science SpecialNeeds': 0.1799163179916318, 'Math_Science Warmth
Care_Hunger': 0.0, 'Music_Arts': 0.12872628726287264, 'Music_Arts AppliedLearning': 1.0,
'Music_Arts Health_Sports': 0.25, 'Music_Arts History_Civics': 0.0, 'Music_Arts SpecialNeeds': 0.0
, 'SpecialNeeds': 0.17504051863857376, 'SpecialNeeds Health_Sports': 0.0, 'SpecialNeeds
Music_Arts': 0.16326530612244897, 'SpecialNeeds Warmth Care_Hunger': 0.2, 'Warmth Care_Hunger':
0.06735751295336788}
{'AppliedLearning': 0.8113948919449901, 'AppliedLearning Health_Sports': 0.8452380952380952,
'AppliedLearning History_Civics': 0.875, 'AppliedLearning Literacy_Language': 0.8685121107266436,
'AppliedLearning Math_Science': 0.8356164383561644, 'AppliedLearning Music_Arts':
0.8210526315789474, 'AppliedLearning SpecialNeeds': 0.7727272727272727, 'AppliedLearning Warmth Ca
re_Hunger': 0.0, 'Health_Sports': 0.8573466476462197, 'Health_Sports AppliedLearning': 0.90625, 'H
ealth_Sports History_Civics': 0.875, 'Health_Sports Literacy_Language': 0.9017857142857143, 'Healt
h_Sports Math_Science': 0.8235294117647058, 'Health_Sports Music_Arts': 0.8076923076923077,
'Health_Sports SpecialNeeds': 0.8920454545454546, 'Health_Sports Warmth Care_Hunger': 1.0,
'History_Civics': 0.831275720164609, 'History_Civics AppliedLearning': 0.5, 'History_Civics
Health_Sports': 1.0, 'History_Civics Literacy_Language': 0.9114583333333334, 'History_Civics
Math_Science': 0.8292682926829268, 'History_Civics Music_Arts': 0.7555555555555555,
'History_Civics SpecialNeeds': 0.7567567567567568, 'Literacy_Language': 0.8655897592197501,
'Literacy_Language AppliedLearning': 0.8536585365853658, 'Literacy_Language Health_Sports': 1.0, '
Literacy_Language History_Civics': 0.8613861386138614, 'Literacy_Language Math_Science':
0.8767605633802817, 'Literacy_Language Music_Arts': 0.8384279475982532, 'Literacy_Language
SpecialNeeds': 0.8589511754068716, 'Literacy_Language Warmth Care_Hunger': 1.0, 'Math_Science':
0.8103595890410958, 'Math_Science AppliedLearning': 0.8057142857142857, 'Math_Science
Health_Sports': 0.8275862068965517, 'Math_Science History_Civics': 0.8441558441558441,
'Math_Science Literacy_Language': 0.8338557993730408, 'Math_Science Music_Arts':
0.8097560975609757, 'Math_Science SpecialNeeds': 0.8200836820083682, 'Math_Science Warmth
Care_Hunger': 1.0, 'Music_Arts': 0.8712737127371274, 'Music_Arts AppliedLearning': 0.0,
'Music_Arts Health_Sports': 0.75, 'Music_Arts History_Civics': 1.0, 'Music_Arts SpecialNeeds': 1.0
, 'SpecialNeeds': 0.8249594813614263, 'SpecialNeeds Health_Sports': 1.0, 'SpecialNeeds
Music_Arts': 0.8367346938775511, 'SpecialNeeds Warmth Care_Hunger': 0.8, 'Warmth Care_Hunger':
0.9326424870466321}
{'AppliedLearning': 0.1643059490084986, 'AppliedLearning Health_Sports': 0.22448979591836735,
'AppliedLearning History_Civics': 0.21052631578947367, 'AppliedLearning Literacy_Language':
0.1650943396226415, 'AppliedLearning Math_Science': 0.15454545454545454, 'AppliedLearning
Music_Arts': 0.14814814814814814, 'AppliedLearning SpecialNeeds': 0.18674698795180722,
'AppliedLearning Warmth Care_Hunger': 0.0, 'Health_Sports': 0.16512820512820514, 'Health_Sports Ap
pliedLearning': 0.0, 'Health_Sports History_Civics': 0.0, 'Health_Sports Literacy_Language': 0.154
92957746478872, 'Health_Sports Math_Science': 0.07692307692307693, 'Health_Sports Music_Arts': 0.2
9411764705882354, 'Health_Sports SpecialNeeds': 0.13138686131386862, 'Health_Sports Warmth
Care_Hunger': 0.0, 'History_Civics': 0.16574585635359115, 'History_Civics AppliedLearning': 0.0, '
History_Civics Health_Sports': 0.0, 'History_Civics Literacy_Language': 0.096, 'History_Civics
Math_Science': 0.08333333333333333, 'History_Civics Music_Arts': 0.10810810810810811,
'History_Civics SpecialNeeds': 0.1111111111111111, 'History_Civics Warmth Care_Hunger': 1.0,
'Literacy_Language': 0.13243847874720358, 'Literacy_Language AppliedLearning':
0.11320754716981132, 'Literacy_Language Health_Sports': 0.0, 'Literacy_Language History_Civics': 0
.16304347826086957, 'Literacy_Language Math_Science': 0.12305516265912306, 'Literacy_Language
Music_Arts': 0.15083798882681565, 'Literacy_Language SpecialNeeds': 0.12912087912087913,

```
'Literacy_Language Warmth Care_Hunger': 0.25, 'Math_Science': 0.17076023391812867, 'Math_Science AppliedLearning': 0.07758620689655173, 'Math_Science Health_Sports': 0.17777777777777778, 'Math_Science History_Civics': 0.1896551724137931, 'Math_Science Literacy_Language': 0.14903846153846154, 'Math_Science Music_Arts': 0.18791946308724833, 'Math_Science SpecialNeeds': 0.19875776397515527, 'Math_Science Warmth Care_Hunger': 0.6666666666666666, 'Music_Arts': 0.1515837104072398, 'Music_Arts AppliedLearning': 0.0, 'Music_Arts Health_Sports': 1.0, 'Music_Arts History_Civics': 1.0, 'Music_Arts SpecialNeeds': 0.16666666666666666, 'SpecialNeeds': 0.22588235294117648, 'SpecialNeeds Health_Sports': 0.25, 'SpecialNeeds Music_Arts': 0.18181818181818182, 'SpecialNeeds Warmth Care_Hunger': 0.5, 'Warmth Care_Hunger': 0.08571428571428572}
{'AppliedLearning': 0.8356940509915014, 'AppliedLearning Health_Sports': 0.7755102040816326, 'AppliedLearning History_Civics': 0.7894736842105263, 'AppliedLearning Literacy_Language': 0.8349056603773585, 'AppliedLearning Math_Science': 0.8454545454545455, 'AppliedLearning Music_Arts': 0.8518518518518519, 'AppliedLearning SpecialNeeds': 0.8132530120481928, 'AppliedLearning Warmth Care_Hunger': 1.0, 'Health_Sports': 0.8348717948717949, 'Health_Sports AppliedLearning': 1.0, 'Health_Sports History_Civics': 1.0, 'Health_Sports Literacy_Language': 0.8450704225352113, 'Health_Sports Math_Science': 0.9230769230769231, 'Health_Sports Music_Arts': 0.7058823529411765, 'Health_Sports SpecialNeeds': 0.8686131386861314, 'Health_Sports Warmth Care_Hunger': 1.0, 'History_Civics': 0.8342541436464088, 'History_Civics AppliedLearning': 1.0, 'History_Civics Health_Sports': 1.0, 'History_Civics Literacy_Language': 0.904, 'History_Civics Math_Science': 0.9166666666666666, 'History_Civics Music_Arts': 0.8918918918918919, 'History_Civics SpecialNeeds': 0.8888888888888888, 'History_Civics Warmth Care_Hunger': 0.0, 'Literacy_Language': 0.8675615212527964, 'Literacy_Language AppliedLearning': 0.8867924528301887, 'Literacy_Language Health_Sports': 1.0, 'Literacy_Language History_Civics': 0.8369565217391305, 'Literacy_Language Math_Science': 0.8769448373408769, 'Literacy_Language Music_Arts': 0.8491620111731844, 'Literacy_Language SpecialNeeds': 0.8708791208791209, 'Literacy_Language Warmth Care_Hunger': 0.75, 'Math_Science': 0.8292397660818713, 'Math_Science AppliedLearning': 0.9224137931034483, 'Math_Science Health_Sports': 0.8222222222222222, 'Math_Science History_Civics': 0.8103448275862069, 'Math_Science Literacy_Language': 0.8509615384615384, 'Math_Science Music_Arts': 0.8120805369127517, 'Math_Science SpecialNeeds': 0.8012422360248447, 'Math_Science Warmth Care_Hunger': 0.3333333333333333, 'Music_Arts': 0.8484162895927602, 'Music_Arts AppliedLearning': 1.0, 'Music_Arts Health_Sports': 0.0, 'Music_Arts History_Civics': 0.0, 'Music_Arts SpecialNeeds': 0.8333333333333334, 'SpecialNeeds': 0.7741176470588236, 'SpecialNeeds Health_Sports': 0.75, 'SpecialNeeds Music_Arts': 0.8181818181818182, 'SpecialNeeds Warmth Care_Hunger': 0.5, 'Warmth Care_Hunger': 0.9142857142857143}
```

Vectorize the Categorical Features - subcategories

In [32]:

```
#Clean SUB Categories
subcat_0_train = get_response(X_train['clean_subcategories'],y_train)[0]
subcat_1_train = get_response(X_train['clean_subcategories'],y_train)[1]

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

subcat_0_cv = get_response(X_cv['clean_subcategories'],y_cv)[0]
subcat_1_cv = get_response(X_cv['clean_subcategories'],y_cv)[1]

print(subcat_0_train)
print(subcat_1_train)

print(subcat_0_test)
print(subcat_1_test)

print(subcat_0_cv)
print(subcat_1_cv)
#===================================add to train df
subcat_neg_train = []
subcat_pos_train = []
for i in X_train['clean_subcategories']:
    subcat_neg_train.append(subcat_0_train[i])
    subcat_pos_train.append(subcat_1_train[i])
X_train['subcat_0'] = subcat_neg_train
X_train['subcat_1'] = subcat_pos_train

#========================================add to test df
subcat_neg_test = []
subcat_pos_test = []
for i in X_test['clean_subcategories']:
    subcat_neg_test.append(subcat_0_test[i])
    subcat_pos_test.append(subcat_1_test[i])
X_test['subcat_0'] = subcat_neg_test
X_test['subcat_1'] = subcat_pos_test
```

```
X_test['subcat_1'] = subcat_pos_test

#=========================================add to cv df
subcat_neg_cv = []
subcat_pos_cv = []
for i in X_cv['clean_subcategories']:
    subcat_neg_cv.append(subcat_0_cv[i])
    subcat_pos_cv.append(subcat_1_cv[i])
X_cv['subcat_0'] = subcat_neg_cv
X_cv['subcat_1'] = subcat_pos_cv
```


```
{'AppliedSciences': 0.1891891891891892, 'AppliedSciences CharacterEducation': 0.2857142857142857,
'AppliedSciences Civics_Government': 0.0, 'AppliedSciences College_CareerPrep':
0.1568627450980392, 'AppliedSciences CommunityService': 0.0, 'AppliedSciences ESL': 0.05,
'AppliedSciences EarlyDevelopment': 0.275, 'AppliedSciences Economics': 0.0, 'AppliedSciences Envi
ronmentalScience': 0.2314410480349345, 'AppliedSciences Extracurricular': 0.10344827586206896, 'Ap
pliedSciences FinancialLiteracy': 0.3333333333333333, 'AppliedSciences Gym_Fitness': 0.0,
'AppliedSciences Health_LifeScience': 0.17164179104477612, 'AppliedSciences Health_Wellness':
0.2857142857142857, 'AppliedSciences History_Geography': 0.2608695652173913, 'AppliedSciences Lite
racy': 0.13836477987421383, 'AppliedSciences Literature_Writing': 0.14285714285714285,
'AppliedSciences Mathematics': 0.17786561264822134, 'AppliedSciences Music': 0.21428571428571427,
'AppliedSciences NutritionEducation': 0.0, 'AppliedSciences Other': 0.18421052631578946,
'AppliedSciences ParentInvolvement': 0.13333333333333333, 'AppliedSciences PerformingArts': 0.0, '
AppliedSciences SocialSciences': 0.13333333333333333, 'AppliedSciences SpecialNeeds':
0.16091954022988506, 'AppliedSciences TeamSports': 0.0, 'AppliedSciences VisualArts':
0.13939393939393940, 'AppliedSciences Warmth Care_Hunger': 1.0, 'CharacterEducation':
0.2878787878787879, 'CharacterEducation Civics_Government': 0.5, 'CharacterEducation
College_CareerPrep': 0.2222222222222222, 'CharacterEducation CommunityService':
0.17647058823529413, 'CharacterEducation ESL': 0.25, 'CharacterEducation EarlyDevelopment': 0.2, '
CharacterEducation EnvironmentalScience': 0.0, 'CharacterEducation Extracurricular': 0.0,
'CharacterEducation ForeignLanguages': 0.3333333333333333, 'CharacterEducation Gym_Fitness':
0.3333333333333333, 'CharacterEducation Health_LifeScience': 0.0, 'CharacterEducation
Health_Wellness': 0.24, 'CharacterEducation History_Geography': 0.0, 'CharacterEducation
Literacy': 0.09523809523809523, 'CharacterEducation Literature_Writing': 0.1, 'CharacterEducation
Mathematics': 0.17391304347826086, 'CharacterEducation Music': 0.25, 'CharacterEducation
NutritionEducation': 0.0, 'CharacterEducation Other': 0.24, 'CharacterEducation
ParentInvolvement': 0.0, 'CharacterEducation PerformingArts': 0.5, 'CharacterEducation
SocialSciences': 0.2, 'CharacterEducation SpecialNeeds': 0.09090909090909091, 'CharacterEducation
TeamSports': 0.5555555555555556, 'CharacterEducation VisualArts': 0.16666666666666666,
'CharacterEducation Warmth Care_Hunger': 1.0, 'Civics_Government': 0.05555555555555555,
'Civics_Government College_CareerPrep': 0.0, 'Civics_Government CommunityService': 0.0,
'Civics_Government ESL': 0.0, 'Civics_Government Economics': 0.3333333333333333,
'Civics_Government EnvironmentalScience': 0.16666666666666666, 'Civics_Government
FinancialLiteracy': 0.0, 'Civics_Government ForeignLanguages': 0.0, 'Civics_Government
Health_LifeScience': 0.0, 'Civics_Government History_Geography': 0.19672131147540983,
'Civics_Government Literacy': 0.03125, 'Civics_Government Literature_Writing':
0.23809523809523808, 'Civics_Government Mathematics': 0.0, 'Civics_Government ParentInvolvement':
0.0, 'Civics_Government PerformingArts': 0.5, 'Civics_Government SocialSciences':
0.13636363636363635, 'Civics_Government SpecialNeeds': 0.0, 'Civics_Government VisualArts': 0.5, '
College_CareerPrep': 0.2159090909090909, 'College_CareerPrep CommunityService': 0.0,
'College_CareerPrep ESL': 0.5, 'College_CareerPrep EarlyDevelopment': 0.2857142857142857,
'College_CareerPrep Economics': 0.0, 'College_CareerPrep EnvironmentalScience': 0.375,
'College_CareerPrep Extracurricular': 0.0, 'College_CareerPrep FinancialLiteracy': 0.0,
'College_CareerPrep ForeignLanguages': 0.0, 'College_CareerPrep Health_LifeScience': 0.0,
'College_CareerPrep Health_Wellness': 0.16666666666666666, 'College_CareerPrep History_Geography':
0.0, 'College_CareerPrep Literacy': 0.12, 'College_CareerPrep Literature_Writing':
0.11594202898550725, 'College_CareerPrep Mathematics': 0.20512820512820512, 'College_CareerPrep Mu
sic': 0.0, 'College_CareerPrep Other': 0.2, 'College_CareerPrep ParentInvolvement':
0.3333333333333333, 'College_CareerPrep PerformingArts': 0.16666666666666666, 'College_CareerPrep
SocialSciences': 0.0, 'College_CareerPrep SpecialNeeds': 0.14285714285714285, 'College_CareerPrep
TeamSports': 0.0, 'College_CareerPrep VisualArts': 0.2608695652173913, 'CommunityService':
0.2631578947368421, 'CommunityService EnvironmentalScience': 0.23076923076923078,
'CommunityService Extracurricular': 1.0, 'CommunityService Health_LifeScience': 0.0,
'CommunityService Health_Wellness': 0.5, 'CommunityService History_Geography': 0.0,
'CommunityService Literacy': 0.0, 'CommunityService Literature_Writing': 0.14285714285714285,
'CommunityService Mathematics': 0.0, 'CommunityService NutritionEducation': 1.0, 'CommunityService
Other': 0.0, 'CommunityService ParentInvolvement': 0.5, 'CommunityService SocialSciences': 0.0, 'C
ommunityService SpecialNeeds': 0.4, 'CommunityService VisualArts': 0.3, 'ESL':
0.19753086419753085, 'ESL EarlyDevelopment': 0.13043478260869565, 'ESL EnvironmentalScience': 0.2,
'ESL Extracurricular': 0.0, 'ESL FinancialLiteracy': 0.5, 'ESL ForeignLanguages': 0.125, 'ESL Gym_
Fitness': 1.0, 'ESL Health_LifeScience': 0.08333333333333333, 'ESL Health_Wellness': 0.0, 'ESL His
tory_Geography': 0.0, 'ESL Literacy': 0.1090146750524109, 'ESL Literature_Writing':
0.15723270440251572, 'ESL Mathematics': 0.2, 'ESL Music': 0.0, 'ESL Other': 0.0, 'ESL
ParentInvolvement': 0.5, 'ESL PerformingArts': 0.2, 'ESL SocialSciences': 0.16666666666666666, 'ES
L SpecialNeeds': 0.17142857142857143, 'ESL TeamSports': 0.0, 'ESL VisualArts': 0.5,
'EarlyDevelopment': 0.19047619047619047, 'EarlyDevelopment EnvironmentalScience':
0.08333333333333333, 'EarlyDevelopment Extracurricular': 0.0, 'EarlyDevelopment Gym_Fitness':
```

0.09090909090909091, 'EarlyDevelopment Health_LifeScience': 0.0, 'EarlyDevelopment
Health_Wellness': 0.09722222222222222, 'EarlyDevelopment Literacy': 0.1377245508982036,
'EarlyDevelopment Literature_Writing': 0.1590909090909091, 'EarlyDevelopment Mathematics':
0.1568627450980392, 'EarlyDevelopment Music': 0.6666666666666666, 'EarlyDevelopment
NutritionEducation': 0.0, 'EarlyDevelopment Other': 0.0967741935483871, 'EarlyDevelopment
ParentInvolvement': 0.3333333333333333, 'EarlyDevelopment PerformingArts': 0.2, 'EarlyDevelopment
SocialSciences': 0.0, 'EarlyDevelopment SpecialNeeds': 0.18421052631578946, 'EarlyDevelopment
TeamSports': 0.0, 'EarlyDevelopment VisualArts': 0.35135135135135137, 'EarlyDevelopment Warmth Car
e_Hunger': 0.0, 'Economics': 0.14285714285714285, 'Economics EnvironmentalScience': 0.0,
'Economics FinancialLiteracy': 0.25, 'Economics History_Geography': 0.3333333333333333, 'Economics
Literacy': 0.0, 'Economics Literature_Writing': 0.0, 'Economics Mathematics': 0.0, 'Economics
Music': 0.0, 'Economics SocialSciences': 0.25, 'Economics VisualArts': 0.0,
'EnvironmentalScience': 0.19008264462809918, 'EnvironmentalScience Extracurricular': 0.0,
'EnvironmentalScience FinancialLiteracy': 0.0, 'EnvironmentalScience ForeignLanguages':
0.3333333333333333, 'EnvironmentalScience Gym_Fitness': 0.0, 'EnvironmentalScience
Health_LifeScience': 0.19289340101522842, 'EnvironmentalScience Health_Wellness': 0.1,
'EnvironmentalScience History_Geography': 0.0967741935483871, 'EnvironmentalScience Literacy': 0.1
0869565217391304, 'EnvironmentalScience Literature_Writing': 0.13333333333333333,
'EnvironmentalScience Mathematics': 0.13872832369942195, 'EnvironmentalScience
NutritionEducation': 0.5, 'EnvironmentalScience Other': 0.5, 'EnvironmentalScience
ParentInvolvement': 0.3333333333333333, 'EnvironmentalScience PerformingArts':
0.16666666666666666, 'EnvironmentalScience SocialSciences': 0.16666666666666666,
'EnvironmentalScience SpecialNeeds': 0.03125, 'EnvironmentalScience VisualArts':
0.1509433962264151, 'Extracurricular': 0.23076923076923078, 'Extracurricular ForeignLanguages':
0.0, 'Extracurricular Gym_Fitness': 0.0, 'Extracurricular Health_LifeScience': 0.5,
'Extracurricular Health_Wellness': 0.5, 'Extracurricular Literacy': 0.2857142857142857,
'Extracurricular Literature_Writing': 0.0, 'Extracurricular Mathematics': 0.25, 'Extracurricular M
usic': 0.25, 'Extracurricular NutritionEducation': 1.0, 'Extracurricular Other':
0.14285714285714285, 'Extracurricular ParentInvolvement': 0.25, 'Extracurricular PerformingArts':
0.25, 'Extracurricular SocialSciences': 0.0, 'Extracurricular SpecialNeeds': 0.0, 'Extracurricular
TeamSports': 0.2222222222222222, 'Extracurricular VisualArts': 0.2, 'FinancialLiteracy':
0.11904761904761904, 'FinancialLiteracy Health_LifeScience': 0.0, 'FinancialLiteracy
History_Geography': 0.0, 'FinancialLiteracy Literacy': 0.125, 'FinancialLiteracy
Literature_Writing': 0.0, 'FinancialLiteracy Mathematics': 0.08823529411764706, 'FinancialLiteracy
ParentInvolvement': 0.0, 'FinancialLiteracy SpecialNeeds': 0.07692307692307693,
'ForeignLanguages': 0.12643678160919541, 'ForeignLanguages Health_LifeScience': 0.0,
'ForeignLanguages History_Geography': 0.0, 'ForeignLanguages Literacy': 0.19148936170212766,
'ForeignLanguages Literature_Writing': 0.16666666666666666, 'ForeignLanguages Mathematics':
0.16666666666666666, 'ForeignLanguages Music': 0.6666666666666666, 'ForeignLanguages Other': 0.0,
'ForeignLanguages SocialSciences': 0.0, 'ForeignLanguages SpecialNeeds': 0.0, 'ForeignLanguages Vi
sualArts': 0.5, 'Gym_Fitness': 0.1598360655737705, 'Gym_Fitness Health_LifeScience': 0.0,
'Gym_Fitness Health_Wellness': 0.11909650924024641, 'Gym_Fitness History_Geography': 0.0,
'Gym_Fitness Literacy': 0.36363636363636365, 'Gym_Fitness Literature_Writing': 0.3333333333333333,
'Gym_Fitness Mathematics': 0.1111111111111111, 'Gym_Fitness Music': 0.0, 'Gym_Fitness
NutritionEducation': 0.1111111111111111, 'Gym_Fitness SpecialNeeds': 0.09302325581395349,
'Gym_Fitness TeamSports': 0.2421875, 'Gym_Fitness VisualArts': 0.3333333333333333,
'Health_LifeScience': 0.12698412698412698, 'Health_LifeScience Health_Wellness':
0.15151515151515152, 'Health_LifeScience History_Geography': 0.21428571428571427,
'Health_LifeScience Literacy': 0.136986301369863, 'Health_LifeScience Literature_Writing':
0.15384615384615385, 'Health_LifeScience Mathematics': 0.17857142857142858, 'Health_LifeScience Mu
sic': 0.0, 'Health_LifeScience NutritionEducation': 0.18181818181818182, 'Health_LifeScience
Other': 0.25, 'Health_LifeScience ParentInvolvement': 0.0, 'Health_LifeScience SocialSciences': 0.
15789473684210525, 'Health_LifeScience SpecialNeeds': 0.22857142857142856, 'Health_LifeScience Tea
mSports': 0.6666666666666666, 'Health_LifeScience VisualArts': 0.35294117647058826,
'Health_Wellness': 0.14388489208633093, 'Health_Wellness History_Geography': 0.4, 'Health_Wellness
Literacy': 0.19444444444444445, 'Health_Wellness Literature_Writing': 0.2054794520547945,
'Health_Wellness Mathematics': 0.125, 'Health_Wellness Music': 0.2, 'Health_Wellness
NutritionEducation': 0.16574585635359115, 'Health_Wellness Other': 0.14285714285714285,
'Health_Wellness PerformingArts': 0.25, 'Health_Wellness SocialSciences': 0.25, 'Health_Wellness S
pecialNeeds': 0.125, 'Health_Wellness TeamSports': 0.2236842105263158, 'Health_Wellness
VisualArts': 0.25, 'Health_Wellness Warmth Care_Hunger': 0.0, 'History_Geography':
0.10256410256410256, 'History_Geography Literacy': 0.06611570247933884, 'History_Geography
Literature_Writing': 0.11764705882352941, 'History_Geography Mathematics': 0.18518518518518517,
'History_Geography Music': 0.0, 'History_Geography Other': 0.3333333333333333, 'History_Geography
PerformingArts': 0.0, 'History_Geography SocialSciences': 0.1527777777777778, 'History_Geography S
pecialNeeds': 0.26666666666666666, 'History_Geography TeamSports': 0.5, 'History_Geography
VisualArts': 0.0625, 'Literacy': 0.12296983758700696, 'Literacy Literature_Writing':
0.12283594394064304, 'Literacy Mathematics': 0.12008390141583639, 'Literacy Music':
0.09090909090909091, 'Literacy NutritionEducation': 0.5, 'Literacy Other': 0.13333333333333333, 'L
iteracy ParentInvolvement': 0.1724137931034483, 'Literacy PerformingArts': 0.03333333333333333, 'L
iteracy SocialSciences': 0.11538461538461539, 'Literacy SpecialNeeds': 0.1407678244972578,
'Literacy TeamSports': 0.25, 'Literacy VisualArts': 0.18181818181818182, 'Literature_Writing': 0.1
4341846758349705, 'Literature_Writing Mathematics': 0.1293233082706767, 'Literature_Writing
Music': 0.07142857142857142, 'Literature_Writing Other': 0.125, 'Literature_Writing
ParentInvolvement': 0.2222222222222222, 'Literature_Writing PerformingArts': 0.15384615384615385,
'Literature_Writing SocialSciences': 0.13157894736842105, 'Literature_Writing SpecialNeeds':
0.2117263843648208, 'Literature_Writing TeamSports': 0.0, 'Literature_Writing VisualArts':

0.1527777777777778, 'Literature_Writing Warmth Care_Hunger': 0.0, 'Mathematics':
0.1950207468879668, 'Mathematics Music': 0.058823529411764705, 'Mathematics NutritionEducation': 0
.3333333333333333, 'Mathematics Other': 0.21052631578947367, 'Mathematics ParentInvolvement':
0.1111111111111111, 'Mathematics PerformingArts': 0.16666666666666666, 'Mathematics
SocialSciences': 0.17391304347826086, 'Mathematics SpecialNeeds': 0.18996415770609318,
'Mathematics TeamSports': 0.5, 'Mathematics VisualArts': 0.12371134020618557, 'Music':
0.10094637223974763, 'Music ParentInvolvement': 0.0, 'Music PerformingArts': 0.11965811965811966,
'Music SocialSciences': 0.5, 'Music SpecialNeeds': 0.1111111111111111, 'Music TeamSports': 0.0, 'M
usic VisualArts': 0.45454545454545453, 'NutritionEducation': 0.1935483870967742,
'NutritionEducation Other': 0.0, 'NutritionEducation SpecialNeeds': 0.25, 'NutritionEducation
TeamSports': 0.0, 'NutritionEducation VisualArts': 0.0, 'NutritionEducation Warmth Care_Hunger': 0
.0, 'Other': 0.164319248826291l, 'Other ParentInvolvement': 0.0, 'Other SocialSciences': 0.0, 'Oth
er SpecialNeeds': 0.23076923076923078, 'Other TeamSports': 0.0, 'Other VisualArts':
0.36363636363636365, 'ParentInvolvement': 0.25, 'ParentInvolvement SocialSciences': 0.0,
'ParentInvolvement SpecialNeeds': 0.2, 'ParentInvolvement VisualArts': 0.2857142857142857,
'PerformingArts': 0.125, 'PerformingArts SocialSciences': 0.25, 'PerformingArts SpecialNeeds': 0.5
, 'PerformingArts VisualArts': 0.25, 'SocialSciences': 0.275, 'SocialSciences SpecialNeeds':
0.15384615384615385, 'SocialSciences VisualArts': 0.08333333333333333, 'SpecialNeeds':
0.1696712619300106l, 'SpecialNeeds TeamSports': 0.2222222222222222, 'SpecialNeeds VisualArts':
0.21311475409836064, 'SpecialNeeds Warmth Care_Hunger': 0.3333333333333333, 'TeamSports':
0.2073170731707317, 'VisualArts': 0.1593625498007968, 'VisualArts Warmth Care_Hunger': 0.5,
'Warmth Care_Hunger': 0.06643356643356643}
{'AppliedSciences': 0.8108108108108109, 'AppliedSciences CharacterEducation': 0.7142857142857143,
'AppliedSciences Civics_Government': 1.0, 'AppliedSciences College_CareerPrep':
0.8431372549019608, 'AppliedSciences CommunityService': 1.0, 'AppliedSciences ESL': 0.95,
'AppliedSciences EarlyDevelopment': 0.725, 'AppliedSciences Economics': 1.0, 'AppliedSciences Envi
ronmentalScience': 0.7685589519650655, 'AppliedSciences Extracurricular': 0.896551724137931,
'AppliedSciences FinancialLiteracy': 0.6666666666666666, 'AppliedSciences Gym_Fitness': 1.0,
'AppliedSciences Health_LifeScience': 0.8283582089552238, 'AppliedSciences Health_Wellness':
0.7142857142857143, 'AppliedSciences History_Geography': 0.7391304347826086, 'AppliedSciences Lite
racy': 0.8616352201257862, 'AppliedSciences Literature_Writing': 0.8571428571428571,
'AppliedSciences Mathematics': 0.8221343873517787, 'AppliedSciences Music': 0.7857142857142857, 'A
ppliedSciences NutritionEducation': 1.0, 'AppliedSciences Other': 0.8157894736842105,
'AppliedSciences ParentInvolvement': 0.8666666666666667, 'AppliedSciences PerformingArts': 1.0, 'A
ppliedSciences SocialSciences': 0.8666666666666667, 'AppliedSciences SpecialNeeds':
0.8390804597701149, 'AppliedSciences TeamSports': 1.0, 'AppliedSciences VisualArts':
0.8606060606060606, 'AppliedSciences Warmth Care_Hunger': 0.0, 'CharacterEducation':
0.7121212121212122, 'CharacterEducation Civics_Government': 0.5, 'CharacterEducation
College_CareerPrep': 0.7777777777777778, 'CharacterEducation CommunityService':
0.8235294117647058, 'CharacterEducation ESL': 0.75, 'CharacterEducation EarlyDevelopment': 0.8,
'CharacterEducation EnvironmentalScience': 1.0, 'CharacterEducation Extracurricular': 1.0,
'CharacterEducation ForeignLanguages': 0.6666666666666666, 'CharacterEducation Gym_Fitness':
0.6666666666666666, 'CharacterEducation Health_LifeScience': 1.0, 'CharacterEducation
Health_Wellness': 0.76, 'CharacterEducation History_Geography': 1.0, 'CharacterEducation
Literacy': 0.9047619047619048, 'CharacterEducation Literature_Writing': 0.9, 'CharacterEducation M
athematics': 0.8260869565217391, 'CharacterEducation Music': 0.75, 'CharacterEducation
NutritionEducation': 1.0, 'CharacterEducation Other': 0.76, 'CharacterEducation
ParentInvolvement': 1.0, 'CharacterEducation PerformingArts': 0.5, 'CharacterEducation
SocialSciences': 0.8, 'CharacterEducation SpecialNeeds': 0.9090909090909091, 'CharacterEducation T
eamSports': 0.4444444444444444, 'CharacterEducation VisualArts': 0.8333333333333334,
'CharacterEducation Warmth Care_Hunger': 0.0, 'Civics_Government': 0.9444444444444444,
'Civics_Government College_CareerPrep': 1.0, 'Civics_Government CommunityService': 1.0,
'Civics_Government ESL': 1.0, 'Civics_Government Economics': 0.6666666666666666,
'Civics_Government EnvironmentalScience': 0.8333333333333334, 'Civics_Government
FinancialLiteracy': 1.0, 'Civics_Government ForeignLanguages': 1.0, 'Civics_Government
Health_LifeScience': 1.0, 'Civics_Government History_Geography': 0.8032786885245902,
'Civics_Government Literacy': 0.96875, 'Civics_Government Literature_Writing': 0.7619047619047619,
'Civics_Government Mathematics': 1.0, 'Civics_Government ParentInvolvement': 1.0,
'Civics_Government PerformingArts': 0.5, 'Civics_Government SocialSciences': 0.8636363636363636, '
Civics_Government SpecialNeeds': 1.0, 'Civics_Government VisualArts': 0.5, 'College_CareerPrep': 0
.7840909090909091, 'College_CareerPrep CommunityService': 1.0, 'College_CareerPrep ESL': 0.5,
'College_CareerPrep EarlyDevelopment': 0.7142857142857143, 'College_CareerPrep Economics': 1.0, 'C
ollege_CareerPrep EnvironmentalScience': 0.625, 'College_CareerPrep Extracurricular': 1.0,
'College_CareerPrep FinancialLiteracy': 1.0, 'College_CareerPrep ForeignLanguages': 1.0,
'College_CareerPrep Health_LifeScience': 1.0, 'College_CareerPrep Health_Wellness':
0.8333333333333334, 'College_CareerPrep History_Geography': 1.0, 'College_CareerPrep Literacy': 0.
88, 'College_CareerPrep Literature_Writing': 0.8840579710144928, 'College_CareerPrep Mathematics':
0.7948717948717948, 'College_CareerPrep Music': 1.0, 'College_CareerPrep Other': 0.8,
'College_CareerPrep ParentInvolvement': 0.6666666666666666, 'College_CareerPrep PerformingArts': 0
.8333333333333334, 'College_CareerPrep SocialSciences': 1.0, 'College_CareerPrep SpecialNeeds': 0.
8571428571428571, 'College_CareerPrep TeamSports': 1.0, 'College_CareerPrep VisualArts':
0.7391304347826086, 'CommunityService': 0.7368421052631579, 'CommunityService
EnvironmentalScience': 0.7692307692307693, 'CommunityService Extracurricular': 0.0,
'CommunityService Health_LifeScience': 1.0, 'CommunityService Health_Wellness': 0.5,
'CommunityService History_Geography': 1.0, 'CommunityService Literacy': 1.0, 'CommunityService Lit
erature_Writing': 0.8571428571428571, 'CommunityService Mathematics': 1.0, 'CommunityService
NutritionEducation': 0.0, 'CommunityService Other': 1.0, 'CommunityService ParentInvolvement': 0.5

, 'CommunityService SocialSciences': 1.0, 'CommunityService SpecialNeeds': 0.6, 'CommunityService
VisualArts': 0.7, 'ESL': 0.8024691358024691, 'ESL EarlyDevelopment': 0.8695652173913043, 'ESL Envi
ronmentalScience': 0.8, 'ESL Extracurricular': 1.0, 'ESL FinancialLiteracy': 0.5, 'ESL
ForeignLanguages': 0.875, 'ESL Gym_Fitness': 0.0, 'ESL Health_LifeScience': 0.9166666666666666, 'E
SL Health_Wellness': 1.0, 'ESL History_Geography': 1.0, 'ESL Literacy': 0.8909853249475891, 'ESL L
iterature_Writing': 0.8427672955974843, 'ESL Mathematics': 0.8, 'ESL Music': 1.0, 'ESL Other':
1.0, 'ESL ParentInvolvement': 0.5, 'ESL PerformingArts': 0.8, 'ESL SocialSciences':
0.8333333333333334, 'ESL SpecialNeeds': 0.8285714285714286, 'ESL TeamSports': 1.0, 'ESL
VisualArts': 0.5, 'EarlyDevelopment': 0.8095238095238095, 'EarlyDevelopment EnvironmentalScience':
0.9166666666666666, 'EarlyDevelopment Extracurricular': 1.0, 'EarlyDevelopment Gym_Fitness':
0.9090909090909091, 'EarlyDevelopment Health_LifeScience': 1.0, 'EarlyDevelopment
Health_Wellness': 0.9027777777777778, 'EarlyDevelopment Literacy': 0.8622754491017964,
'EarlyDevelopment Literature_Writing': 0.8409090909090909, 'EarlyDevelopment Mathematics':
0.8431372549019608, 'EarlyDevelopment Music': 0.3333333333333333, 'EarlyDevelopment
NutritionEducation': 1.0, 'EarlyDevelopment Other': 0.9032258064516129, 'EarlyDevelopment
ParentInvolvement': 0.6666666666666666, 'EarlyDevelopment PerformingArts': 0.8, 'EarlyDevelopment
SocialSciences': 1.0, 'EarlyDevelopment SpecialNeeds': 0.8157894736842105, 'EarlyDevelopment
TeamSports': 1.0, 'EarlyDevelopment VisualArts': 0.6486486486486487, 'EarlyDevelopment Warmth Care
_Hunger': 1.0, 'Economics': 0.8571428571428571, 'Economics EnvironmentalScience': 1.0, 'Economics
FinancialLiteracy': 0.75, 'Economics History_Geography': 0.6666666666666666, 'Economics Literacy':
1.0, 'Economics Literature_Writing': 1.0, 'Economics Mathematics': 1.0, 'Economics Music': 1.0,
'Economics SocialSciences': 0.75, 'Economics VisualArts': 1.0, 'EnvironmentalScience':
0.8099173553719008, 'EnvironmentalScience Extracurricular': 1.0, 'EnvironmentalScience
FinancialLiteracy': 1.0, 'EnvironmentalScience ForeignLanguages': 0.6666666666666666,
'EnvironmentalScience Gym_Fitness': 1.0, 'EnvironmentalScience Health_LifeScience':
0.8071065989847716, 'EnvironmentalScience Health_Wellness': 0.9, 'EnvironmentalScience
History_Geography': 0.9032258064516129, 'EnvironmentalScience Literacy': 0.8913043478260869,
'EnvironmentalScience Literature_Writing': 0.8666666666666667, 'EnvironmentalScience Mathematics':
0.861271676300578, 'EnvironmentalScience NutritionEducation': 0.5, 'EnvironmentalScience Other': 0
.5, 'EnvironmentalScience ParentInvolvement': 0.6666666666666666, 'EnvironmentalScience
PerformingArts': 0.8333333333333334, 'EnvironmentalScience SocialSciences': 0.8333333333333334, 'E
nvironmentalScience SpecialNeeds': 0.96875, 'EnvironmentalScience VisualArts': 0.8490566037735849,
'Extracurricular': 0.7692307692307693, 'Extracurricular ForeignLanguages': 1.0, 'Extracurricular G
ym_Fitness': 1.0, 'Extracurricular Health_LifeScience': 0.5, 'Extracurricular Health_Wellness': 0.
5, 'Extracurricular Literacy': 0.7142857142857143, 'Extracurricular Literature_Writing': 1.0, 'Ext
racurricular Mathematics': 0.75, 'Extracurricular Music': 0.75, 'Extracurricular
NutritionEducation': 0.0, 'Extracurricular Other': 0.8571428571428571, 'Extracurricular
ParentInvolvement': 0.75, 'Extracurricular PerformingArts': 0.75, 'Extracurricular
SocialSciences': 1.0, 'Extracurricular SpecialNeeds': 1.0, 'Extracurricular TeamSports':
0.7777777777777778, 'Extracurricular VisualArts': 0.8, 'FinancialLiteracy': 0.8809523809523809, 'F
inancialLiteracy Health_LifeScience': 1.0, 'FinancialLiteracy History_Geography': 1.0,
'FinancialLiteracy Literacy': 0.875, 'FinancialLiteracy Literature_Writing': 1.0,
'FinancialLiteracy Mathematics': 0.9117647058823529, 'FinancialLiteracy ParentInvolvement': 1.0, '
FinancialLiteracy SpecialNeeds': 0.9230769230769231, 'ForeignLanguages': 0.8735632183908046, 'Fore
ignLanguages Health_LifeScience': 1.0, 'ForeignLanguages History_Geography': 1.0,
'ForeignLanguages Literacy': 0.8085106382978723, 'ForeignLanguages Literature_Writing':
0.8333333333333334, 'ForeignLanguages Mathematics': 0.8333333333333334, 'ForeignLanguages Music':
0.3333333333333333, 'ForeignLanguages Other': 1.0, 'ForeignLanguages SocialSciences': 1.0,
'ForeignLanguages SpecialNeeds': 1.0, 'ForeignLanguages VisualArts': 0.5, 'Gym_Fitness':
0.8401639344262295, 'Gym_Fitness Health_LifeScience': 1.0, 'Gym_Fitness Health_Wellness':
0.8809034907597536, 'Gym_Fitness History_Geography': 1.0, 'Gym_Fitness Literacy':
0.6363636363636364, 'Gym_Fitness Literature_Writing': 0.6666666666666666, 'Gym_Fitness
Mathematics': 0.8888888888888888, 'Gym_Fitness Music': 1.0, 'Gym_Fitness NutritionEducation':
0.8888888888888888, 'Gym_Fitness SpecialNeeds': 0.9069767441860465, 'Gym_Fitness TeamSports':
0.7578125, 'Gym_Fitness VisualArts': 0.6666666666666666, 'Health_LifeScience': 0.873015873015873,
'Health_LifeScience Health_Wellness': 0.8484848484848485, 'Health_LifeScience History_Geography':
0.7857142857142857, 'Health_LifeScience Literacy': 0.863013698630137, 'Health_LifeScience
Literature_Writing': 0.8461538461538461, 'Health_LifeScience Mathematics': 0.8214285714285714, 'He
alth_LifeScience Music': 1.0, 'Health_LifeScience NutritionEducation': 0.8181818181818182,
'Health_LifeScience Other': 0.75, 'Health_LifeScience ParentInvolvement': 1.0, 'Health_LifeScience
SocialSciences': 0.8421052631578947, 'Health_LifeScience SpecialNeeds': 0.7714285714285715,
'Health_LifeScience TeamSports': 0.3333333333333333, 'Health_LifeScience VisualArts':
0.6470588235294118, 'Health_Wellness': 0.8561151079136691, 'Health_Wellness History_Geography': 0.
6, 'Health_Wellness Literacy': 0.8055555555555556, 'Health_Wellness Literature_Writing':
0.7945205479452054, 'Health_Wellness Mathematics': 0.875, 'Health_Wellness Music': 0.8,
'Health_Wellness NutritionEducation': 0.8342541436464088, 'Health_Wellness Other':
0.8571428571428571, 'Health_Wellness PerformingArts': 0.75, 'Health_Wellness SocialSciences': 0.75
, 'Health_Wellness SpecialNeeds': 0.875, 'Health_Wellness TeamSports': 0.7763157894736842,
'Health_Wellness VisualArts': 0.75, 'Health_Wellness Warmth Care_Hunger': 1.0,
'History_Geography': 0.8974358974358975, 'History_Geography Literacy': 0.9338842975206612,
'History_Geography Literature_Writing': 0.8823529411764706, 'History_Geography Mathematics':
0.8148148148148148, 'History_Geography Music': 1.0, 'History_Geography Other': 0.6666666666666666,
'History_Geography PerformingArts': 1.0, 'History_Geography SocialSciences': 0.8472222222222222, '
History_Geography SpecialNeeds': 0.7333333333333333, 'History_Geography TeamSports': 0.5,
'History_Geography VisualArts': 0.9375, 'Literacy': 0.877030162412993, 'Literacy
Literature_Writing': 0.877164056059357, 'Literacy Mathematics': 0.8799160985841636, 'Literacy
Music': 0.9090909090909091, 'Literacy NutritionEducation': 0.5, 'Literacy Other':

0.8666666666666667, 'Literacy ParentInvolvement': 0.8275862068965517, 'Literacy PerformingArts': 0
.9666666666666667, 'Literacy SocialSciences': 0.8846153846153846, 'Literacy SpecialNeeds':
0.8592321755027422, 'Literacy TeamSports': 0.75, 'Literacy VisualArts': 0.8181818181818182,
'Literature_Writing': 0.8565815324165029, 'Literature_Writing Mathematics': 0.8706766917293233, 'L
iterature_Writing Music': 0.9285714285714286, 'Literature_Writing Other': 0.875,
'Literature_Writing ParentInvolvement': 0.7777777777777778, 'Literature_Writing PerformingArts': 0
.8461538461538461, 'Literature_Writing SocialSciences': 0.868421052631579, 'Literature_Writing Spe
cialNeeds': 0.7882736156351792, 'Literature_Writing TeamSports': 1.0, 'Literature_Writing
VisualArts': 0.8472222222222222, 'Literature_Writing Warmth Care_Hunger': 1.0, 'Mathematics': 0.80
49792531120332, 'Mathematics Music': 0.9411764705882353, 'Mathematics NutritionEducation':
0.6666666666666666, 'Mathematics Other': 0.7894736842105263, 'Mathematics ParentInvolvement':
0.8888888888888888, 'Mathematics PerformingArts': 0.8333333333333334, 'Mathematics
SocialSciences': 0.8260869565217391, 'Mathematics SpecialNeeds': 0.8100358422939068, 'Mathematics
TeamSports': 0.5, 'Mathematics VisualArts': 0.8762886597938144, 'Music': 0.8990536277602523, 'Musi
c ParentInvolvement': 1.0, 'Music PerformingArts': 0.8803418803418803, 'Music SocialSciences': 0.5
, 'Music SpecialNeeds': 0.8888888888888888, 'Music TeamSports': 1.0, 'Music VisualArts':
0.5454545454545454, 'NutritionEducation': 0.8064516129032258, 'NutritionEducation Other': 1.0, 'Nu
tritionEducation SpecialNeeds': 0.75, 'NutritionEducation TeamSports': 1.0, 'NutritionEducation Vi
sualArts': 1.0, 'NutritionEducation Warmth Care_Hunger': 1.0, 'Other': 0.8356807511737089, 'Other
ParentInvolvement': 1.0, 'Other SocialSciences': 1.0, 'Other SpecialNeeds': 0.7692307692307693, 'O
ther TeamSports': 1.0, 'Other VisualArts': 0.6363636363636364, 'ParentInvolvement': 0.75,
'ParentInvolvement SocialSciences': 1.0, 'ParentInvolvement SpecialNeeds': 0.8, 'ParentInvolvement
VisualArts': 0.7142857142857143, 'PerformingArts': 0.875, 'PerformingArts SocialSciences': 0.75, '
PerformingArts SpecialNeeds': 0.5, 'PerformingArts VisualArts': 0.75, 'SocialSciences': 0.725,
'SocialSciences SpecialNeeds': 0.8461538461538461, 'SocialSciences VisualArts':
0.9166666666666666, 'SpecialNeeds': 0.8303287380699894, 'SpecialNeeds TeamSports':
0.7777777777777778, 'SpecialNeeds VisualArts': 0.7868852459016393, 'SpecialNeeds Warmth
Care_Hunger': 0.6666666666666666, 'TeamSports': 0.792682926829268З, 'VisualArts':
0.8406374501992032, 'VisualArts Warmth Care_Hunger': 0.5, 'Warmth Care_Hunger':
0.9335664335664335}
{'AppliedSciences': 0.1981981981981982, 'AppliedSciences CharacterEducation': 0.0,
'AppliedSciences Civics_Government': 0.0, 'AppliedSciences College_CareerPrep':
0.17391304347826086, 'AppliedSciences CommunityService': 0.3333333333333333, 'AppliedSciences ESL'
: 0.1, 'AppliedSciences EarlyDevelopment': 0.25925925925925924, 'AppliedSciences Economics': 0.0,
'AppliedSciences EnvironmentalScience': 0.22602739726027396, 'AppliedSciences Extracurricular': 0.
25, 'AppliedSciences FinancialLiteracy': 0.0, 'AppliedSciences ForeignLanguages': 1.0,
'AppliedSciences Gym_Fitness': 0.0, 'AppliedSciences Health_LifeScience': 0.22535211267605634,
'AppliedSciences Health_Wellness': 0.125, 'AppliedSciences History_Geography':
0.42857142857142855, 'AppliedSciences Literacy': 0.21875, 'AppliedSciences Literature_Writing': 0.
13559322033898305, 'AppliedSciences Mathematics': 0.16810344827586207, 'AppliedSciences Music': 0.
1111111111111111, 'AppliedSciences NutritionEducation': 0.0, 'AppliedSciences Other':
0.06666666666666667, 'AppliedSciences ParentInvolvement': 0.3076923076923077, 'AppliedSciences Per
formingArts': 0.0, 'AppliedSciences SocialSciences': 0.25, 'AppliedSciences SpecialNeeds':
0.08163265306122448, 'AppliedSciences TeamSports': 0.3333333333333333, 'AppliedSciences
VisualArts': 0.19718309859154928, 'CharacterEducation': 0.26785714285714285, 'CharacterEducation C
ollege_CareerPrep': 0.1875, 'CharacterEducation CommunityService': 0.16666666666666666,
'CharacterEducation ESL': 1.0, 'CharacterEducation EarlyDevelopment': 0.21739130434782608,
'CharacterEducation EnvironmentalScience': 0.5, 'CharacterEducation Extracurricular':
0.2857142857142857, 'CharacterEducation ForeignLanguages': 0.0, 'CharacterEducation Gym_Fitness':
0.5, 'CharacterEducation Health_LifeScience': 0.2, 'CharacterEducation Health_Wellness':
0.2916666666666667, 'CharacterEducation History_Geography': 0.5, 'CharacterEducation Literacy': 0.
1111111111111111, 'CharacterEducation Literature_Writing': 0.18518518518518517,
'CharacterEducation Mathematics': 0.3333333333333333, 'CharacterEducation Music': 0.0,
'CharacterEducation Other': 0.1, 'CharacterEducation ParentInvolvement': 0.0, 'CharacterEducation
PerformingArts': 0.0, 'CharacterEducation SocialSciences': 0.0, 'CharacterEducation SpecialNeeds':
0.25925925925925924, 'CharacterEducation TeamSports': 0.0, 'CharacterEducation VisualArts': 0.2, '
CharacterEducation Warmth Care_Hunger': 1.0, 'Civics_Government': 0.3333333333333333,
'Civics_Government College_CareerPrep': 0.5, 'Civics_Government CommunityService': 1.0,
'Civics_Government Economics': 0.75, 'Civics_Government EnvironmentalScience': 0.3333333333333333,
'Civics_Government FinancialLiteracy': 0.0, 'Civics_Government Health_LifeScience': 0.5,
'Civics_Government Health_Wellness': 0.0, 'Civics_Government History_Geography':
0.18518518518518517, 'Civics_Government Literacy': 0.0, 'Civics_Government Literature_Writing':
0.125, 'Civics_Government Mathematics': 0.0, 'Civics_Government NutritionEducation': 0.0,
'Civics_Government PerformingArts': 0.0, 'Civics_Government SocialSciences': 0.16666666666666666,
'Civics_Government SpecialNeeds': 0.0, 'Civics_Government TeamSports': 0.0, 'Civics_Government
VisualArts': 0.3333333333333333, 'College_CareerPrep': 0.19298245614035087, 'College_CareerPrep Co
mmunityService': 0.0, 'College_CareerPrep ESL': 0.0, 'College_CareerPrep EarlyDevelopment': 0.0, '
College_CareerPrep Economics': 0.0, 'College_CareerPrep EnvironmentalScience': 0.375,
'College_CareerPrep Extracurricular': 0.125, 'College_CareerPrep ForeignLanguages':
0.3333333333333333, 'College_CareerPrep Health_LifeScience': 0.0, 'College_CareerPrep
Health_Wellness': 0.0, 'College_CareerPrep History_Geography': 0.0, 'College_CareerPrep Literacy':
0.11428571428571428, 'College_CareerPrep Literature_Writing': 0.10416666666666667,
'College_CareerPrep Mathematics': 0.07142857142857142, 'College_CareerPrep Music': 0.5,
'College_CareerPrep NutritionEducation': 1.0, 'College_CareerPrep Other': 0.4444444444444444,
'College_CareerPrep ParentInvolvement': 0.3333333333333333, 'College_CareerPrep PerformingArts': 0
.6666666666666666, 'College_CareerPrep SocialSciences': 0.0, 'College_CareerPrep SpecialNeeds': 0.
10526315789473684, 'College_CareerPrep TeamSports': 0.0, 'College_CareerPrep VisualArts':

0.15384615384615385, 'CommunityService': 0.14285714285714285, 'CommunityService ESL': 1.0,
'CommunityService EarlyDevelopment': 0.5, 'CommunityService Economics': 0.0, 'CommunityService Env
ironmentalScience': 0.0, 'CommunityService Extracurricular': 0.25, 'CommunityService
FinancialLiteracy': 0.0, 'CommunityService Health_LifeScience': 0.5, 'CommunityService
Health_Wellness': 0.0, 'CommunityService Literacy': 0.0, 'CommunityService Literature_Writing': 0.
25, 'CommunityService Mathematics': 0.0, 'CommunityService Other': 1.0, 'CommunityService
SocialSciences': 0.5, 'CommunityService SpecialNeeds': 0.5, 'CommunityService VisualArts': 0.0, 'E
SL': 0.16393442622950818, 'ESL EarlyDevelopment': 0.0, 'ESL EnvironmentalScience':
0.3333333333333333, 'ESL Extracurricular': 0.0, 'ESL FinancialLiteracy': 1.0, 'ESL
ForeignLanguages': 0.125, 'ESL Gym_Fitness': 0.0, 'ESL Health_LifeScience': 0.0, 'ESL
Health_Wellness': 0.0, 'ESL History_Geography': 0.2857142857142857, 'ESL Literacy':
0.11578947368421053, 'ESL Literature_Writing': 0.1523809523809524, 'ESL Mathematics':
0.1111111111111111, 'ESL Music': 0.0, 'ESL Other': 0.0, 'ESL ParentInvolvement': 1.0, 'ESL
PerformingArts': 1.0, 'ESL SocialSciences': 0.5, 'ESL SpecialNeeds': 0.037037037037037035, 'ESL Vi
sualArts': 0.0, 'EarlyDevelopment': 0.17543859649122806, 'EarlyDevelopment EnvironmentalScience':
0.16666666666666666, 'EarlyDevelopment Extracurricular': 0.0, 'EarlyDevelopment Gym_Fitness': 0.0,
'EarlyDevelopment Health_LifeScience': 0.5, 'EarlyDevelopment Health_Wellness':
0.07692307692307693, 'EarlyDevelopment Literacy': 0.11627906976744186, 'EarlyDevelopment
Literature_Writing': 0.125, 'EarlyDevelopment Mathematics': 0.14634146341463414, 'EarlyDevelopment
Music': 0.0, 'EarlyDevelopment Other': 0.2, 'EarlyDevelopment ParentInvolvement':
0.1111111111111111, 'EarlyDevelopment PerformingArts': 0.0, 'EarlyDevelopment SpecialNeeds':
0.2457627118644068, 'EarlyDevelopment VisualArts': 0.2608695652173913, 'Economics': 0.0,
'Economics FinancialLiteracy': 0.2727272727272727, 'Economics ForeignLanguages': 0.0, 'Economics H
istory_Geography': 0.0, 'Economics Literacy': 0.0, 'Economics Mathematics': 0.0, 'Economics
SocialSciences': 0.0, 'Economics SpecialNeeds': 0.5, 'Economics VisualArts': 0.0,
'EnvironmentalScience': 0.12857142857142856, 'EnvironmentalScience Extracurricular': 1.0,
'EnvironmentalScience Health_LifeScience': 0.2028985507246377, 'EnvironmentalScience
Health_Wellness': 0.18181818181818182, 'EnvironmentalScience History_Geography': 0.0,
'EnvironmentalScience Literacy': 0.14666666666666667, 'EnvironmentalScience Literature_Writing': 0
.3055555555555556, 'EnvironmentalScience Mathematics': 0.21782178217821782, 'EnvironmentalScience
NutritionEducation': 0.2, 'EnvironmentalScience Other': 0.0, 'EnvironmentalScience
ParentInvolvement': 0.0, 'EnvironmentalScience PerformingArts': 0.0, 'EnvironmentalScience
SocialSciences': 0.3333333333333333, 'EnvironmentalScience SpecialNeeds': 0.1,
'EnvironmentalScience TeamSports': 0.0, 'EnvironmentalScience VisualArts': 0.2222222222222222, 'En
vironmentalScience Warmth Care_Hunger': 0.0, 'Extracurricular': 0.11764705882352941,
'Extracurricular Gym_Fitness': 0.0, 'Extracurricular Health_Wellness': 0.25, 'Extracurricular
Literacy': 0.1111111111111111, 'Extracurricular Literature_Writing': 1.0, 'Extracurricular
Mathematics': 0.1111111111111111, 'Extracurricular Music': 0.0, 'Extracurricular Other': 0.6, 'Ext
racurricular PerformingArts': 0.0, 'Extracurricular TeamSports': 0.0, 'Extracurricular
VisualArts': 0.2, 'FinancialLiteracy': 0.1, 'FinancialLiteracy ForeignLanguages': 0.0,
'FinancialLiteracy History_Geography': 0.0, 'FinancialLiteracy Literacy': 0.0, 'FinancialLiteracy
Literature_Writing': 0.0, 'FinancialLiteracy Mathematics': 0.06666666666666667, 'FinancialLiteracy
SpecialNeeds': 0.25, 'FinancialLiteracy VisualArts': 1.0, 'ForeignLanguages': 0.16666666666666666,
'ForeignLanguages Health_Wellness': 0.0, 'ForeignLanguages History_Geography': 0.5,
'ForeignLanguages Literacy': 0.11538461538461539, 'ForeignLanguages Literature_Writing': 0.125, 'F
oreignLanguages Mathematics': 0.25, 'ForeignLanguages Music': 0.0, 'ForeignLanguages
SocialSciences': 0.5, 'ForeignLanguages SpecialNeeds': 0.0, 'ForeignLanguages VisualArts': 0.0, 'G
ym_Fitness': 0.17058823529411765, 'Gym_Fitness Health_Wellness': 0.10491803278688525, 'Gym_Fitness
Literacy': 0.4, 'Gym_Fitness Literature_Writing': 0.5, 'Gym_Fitness Mathematics': 0.4,
'Gym_Fitness Music': 0.0, 'Gym_Fitness NutritionEducation': 0.25, 'Gym_Fitness ParentInvolvement':
0.0, 'Gym_Fitness PerformingArts': 0.0, 'Gym_Fitness SpecialNeeds': 0.15, 'Gym_Fitness
TeamSports': 0.25, 'Gym_Fitness VisualArts': 0.2, 'Health_LifeScience': 0.19827586206896552,
'Health_LifeScience Health_Wellness': 0.05555555555555555, 'Health_LifeScience History_Geography':
0.07692307692307693, 'Health_LifeScience Literacy': 0.08888888888888889, 'Health_LifeScience
Literature_Writing': 0.07142857142857142, 'Health_LifeScience Mathematics': 0.26865671641791045, '
Health_LifeScience Music': 0.0, 'Health_LifeScience NutritionEducation': 0.5, 'Health_LifeScience
Other': 0.0, 'Health_LifeScience ParentInvolvement': 0.0, 'Health_LifeScience PerformingArts': 0.3
333333333333333, 'Health_LifeScience SocialSciences': 0.0, 'Health_LifeScience SpecialNeeds': 0.3,
'Health_LifeScience TeamSports': 0.0, 'Health_LifeScience VisualArts': 0.29411764705882354,
'Health_Wellness': 0.12127236580516898, 'Health_Wellness History_Geography': 0.2, 'Health_Wellness
Literacy': 0.06557377049180328, 'Health_Wellness Literature_Writing': 0.09090909090909091,
'Health_Wellness Mathematics': 0.13793103448275862, 'Health_Wellness Music': 0.3333333333333333, '
Health_Wellness NutritionEducation': 0.14423076923076922, 'Health_Wellness Other': 0.1,
'Health_Wellness PerformingArts': 0.0, 'Health_Wellness SocialSciences': 0.0, 'Health_Wellness Spe
cialNeeds': 0.10596026490066225, 'Health_Wellness TeamSports': 0.1836734693877551,
'Health_Wellness VisualArts': 0.0, 'Health_Wellness Warmth Care_Hunger': 0.0, 'History_Geography':
0.175, 'History_Geography Literacy': 0.07352941176470588, 'History_Geography Literature_Writing':
0.125, 'History_Geography Mathematics': 0.2222222222222222, 'History_Geography Music': 0.0,
'History_Geography ParentInvolvement': 0.0, 'History_Geography PerformingArts':
0.3333333333333333, 'History_Geography SocialSciences': 0.20930232558139536, 'History_Geography
SpecialNeeds': 0.14285714285714285, 'History_Geography TeamSports': 0.0, 'History_Geography
VisualArts': 0.22727272727272727, 'Literacy': 0.11963882618510158, 'Literacy Literature_Writing':
0.138676844783715, 'Literacy Mathematics': 0.12073490813648294, 'Literacy Music':
0.10714285714285714, 'Literacy Other': 0.16, 'Literacy ParentInvolvement': 0.047619047619047616, '
Literacy PerformingArts': 0.2222222222222222, 'Literacy SocialSciences': 0.1111111111111111, 'Lite
racy SpecialNeeds': 0.14204545454545456, 'Literacy VisualArts': 0.1645569620253164, 'Literacy War
mth Care_Hunger': 0.0, 'Literature_Writing': 0.16164817749603805, 'Literature_Writing

mth Care_Hunger': 0.0, 'Literature_Writing': 0.16164817749003865, 'Literature_Writing
Mathematics': 0.12547528517110265, 'Literature_Writing Music': 0.0, 'Literature_Writing Other': 0.
3125, 'Literature_Writing ParentInvolvement': 0.0, 'Literature_Writing PerformingArts':
0.07142857142857142, 'Literature_Writing SocialSciences': 0.05, 'Literature_Writing SpecialNeeds':
0.1569767441860465, 'Literature_Writing VisualArts': 0.20270270270270271, 'Mathematics':
0.18552631578947368, 'Mathematics Music': 0.0, 'Mathematics NutritionEducation': 0.0, 'Mathematics
Other': 0.2222222222222222, 'Mathematics ParentInvolvement': 0.07142857142857142, 'Mathematics Per
formingArts': 0.0, 'Mathematics SocialSciences': 0.1, 'Mathematics SpecialNeeds':
0.20666666666666667, 'Mathematics VisualArts': 0.20689655172413793, 'Music': 0.11320754716981132,
'Music Other': 1.0, 'Music ParentInvolvement': 1.0, 'Music PerformingArts': 0.08396946564885496, '
Music SocialSciences': 0.0, 'Music SpecialNeeds': 0.0, 'Music TeamSports': 0.5, 'Music
VisualArts': 0.0, 'NutritionEducation': 0.16666666666666666, 'NutritionEducation Other': 0.0, 'Nut
ritionEducation SpecialNeeds': 0.0, 'NutritionEducation TeamSports': 0.3333333333333333,
'NutritionEducation VisualArts': 1.0, 'Other': 0.14414414414414414, 'Other ParentInvolvement': 0.0
, 'Other SocialSciences': 0.0, 'Other SpecialNeeds': 0.19230769230769232, 'Other VisualArts': 0.14
285714285714285, 'ParentInvolvement': 0.25, 'ParentInvolvement VisualArts': 0.0, 'PerformingArts':
0.11764705882352941, 'PerformingArts SpecialNeeds': 0.0, 'PerformingArts TeamSports': 0.0,
'PerformingArts VisualArts': 0.15384615384615385, 'SocialSciences': 0.0, 'SocialSciences
SpecialNeeds': 0.4, 'SocialSciences TeamSports': 0.0, 'SocialSciences VisualArts': 0.2,
'SpecialNeeds': 0.17504051863857376, 'SpecialNeeds TeamSports': 0.0, 'SpecialNeeds VisualArts':
0.16326530612244897, 'SpecialNeeds Warmth Care_Hunger': 0.2, 'TeamSports': 0.16666666666666666,
'VisualArts': 0.16181229773462782, 'Warmth Care_Hunger': 0.06735751295336788}
{'AppliedSciences': 0.8018018018018018, 'AppliedSciences CharacterEducation': 1.0,
'AppliedSciences Civics_Government': 1.0, 'AppliedSciences College_CareerPrep':
0.8260869565217391, 'AppliedSciences CommunityService': 0.6666666666666666, 'AppliedSciences ESL':
0.9, 'AppliedSciences EarlyDevelopment': 0.7407407407407407, 'AppliedSciences Economics': 1.0, 'Ap
pliedSciences EnvironmentalScience': 0.773972602739726, 'AppliedSciences Extracurricular': 0.75, '
AppliedSciences FinancialLiteracy': 1.0, 'AppliedSciences ForeignLanguages': 0.0, 'AppliedSciences
Gym_Fitness': 1.0, 'AppliedSciences Health_LifeScience': 0.7746478873239436, 'AppliedSciences Heal
th_Wellness': 0.875, 'AppliedSciences History_Geography': 0.5714285714285714, 'AppliedSciences Lit
eracy': 0.78125, 'AppliedSciences Literature_Writing': 0.864406779661017, 'AppliedSciences
Mathematics': 0.8318965517241379, 'AppliedSciences Music': 0.8888888888888888, 'AppliedSciences Nu
tritionEducation': 1.0, 'AppliedSciences Other': 0.9333333333333333, 'AppliedSciences
ParentInvolvement': 0.6923076923076923, 'AppliedSciences PerformingArts': 1.0, 'AppliedSciences So
cialSciences': 0.75, 'AppliedSciences SpecialNeeds': 0.9183673469387755, 'AppliedSciences
TeamSports': 0.6666666666666666, 'AppliedSciences VisualArts': 0.8028169014084507,
'CharacterEducation': 0.7321428571428571, 'CharacterEducation College_CareerPrep': 0.8125,
'CharacterEducation CommunityService': 0.8333333333333334, 'CharacterEducation ESL': 0.0,
'CharacterEducation EarlyDevelopment': 0.782608695652174, 'CharacterEducation
EnvironmentalScience': 0.5, 'CharacterEducation Extracurricular': 0.7142857142857143,
'CharacterEducation ForeignLanguages': 1.0, 'CharacterEducation Gym_Fitness': 0.5,
'CharacterEducation Health_LifeScience': 0.8, 'CharacterEducation Health_Wellness':
0.7083333333333334, 'CharacterEducation History_Geography': 0.5, 'CharacterEducation Literacy': 0.
8888888888888888, 'CharacterEducation Literature_Writing': 0.8148148148148148, 'CharacterEducation
Mathematics': 0.6666666666666666, 'CharacterEducation Music': 1.0, 'CharacterEducation Other': 0.9
, 'CharacterEducation ParentInvolvement': 1.0, 'CharacterEducation PerformingArts': 1.0,
'CharacterEducation SocialSciences': 1.0, 'CharacterEducation SpecialNeeds': 0.7407407407407407, '
CharacterEducation TeamSports': 1.0, 'CharacterEducation VisualArts': 0.8, 'CharacterEducation War
mth Care_Hunger': 0.0, 'Civics_Government': 0.6666666666666666, 'Civics_Government
College_CareerPrep': 0.5, 'Civics_Government CommunityService': 0.0, 'Civics_Government
Economics': 0.25, 'Civics_Government EnvironmentalScience': 0.6666666666666666, 'Civics_Government
FinancialLiteracy': 1.0, 'Civics_Government Health_LifeScience': 0.5, 'Civics_Government
Health_Wellness': 1.0, 'Civics_Government History_Geography': 0.8148148148148148,
'Civics_Government Literacy': 1.0, 'Civics_Government Literature_Writing': 0.875,
'Civics_Government Mathematics': 1.0, 'Civics_Government NutritionEducation': 1.0,
'Civics_Government PerformingArts': 1.0, 'Civics_Government SocialSciences': 0.8333333333333334, '
Civics_Government SpecialNeeds': 1.0, 'Civics_Government TeamSports': 1.0, 'Civics_Government
VisualArts': 0.6666666666666666, 'College_CareerPrep': 0.8070175438596491, 'College_CareerPrep Com
munityService': 1.0, 'College_CareerPrep ESL': 1.0, 'College_CareerPrep EarlyDevelopment': 1.0, 'C
ollege_CareerPrep Economics': 1.0, 'College_CareerPrep EnvironmentalScience': 0.625,
'College_CareerPrep Extracurricular': 0.875, 'College_CareerPrep ForeignLanguages':
0.6666666666666666, 'College_CareerPrep Health_LifeScience': 1.0, 'College_CareerPrep
Health_Wellness': 1.0, 'College_CareerPrep History_Geography': 1.0, 'College_CareerPrep Literacy':
0.8857142857142857, 'College_CareerPrep Literature_Writing': 0.8958333333333334,
'College_CareerPrep Mathematics': 0.9285714285714286, 'College_CareerPrep Music': 0.5,
'College_CareerPrep NutritionEducation': 0.0, 'College_CareerPrep Other': 0.5555555555555556,
'College_CareerPrep ParentInvolvement': 0.6666666666666666, 'College_CareerPrep PerformingArts': 0
.3333333333333333, 'College_CareerPrep SocialSciences': 1.0, 'College_CareerPrep SpecialNeeds': 0.
8947368421052632, 'College_CareerPrep TeamSports': 1.0, 'College_CareerPrep VisualArts':
0.8461538461538461, 'CommunityService': 0.8571428571428571, 'CommunityService ESL': 0.0,
'CommunityService EarlyDevelopment': 0.5, 'CommunityService Economics': 1.0, 'CommunityService Env
ironmentalScience': 1.0, 'CommunityService Extracurricular': 0.75, 'CommunityService
FinancialLiteracy': 1.0, 'CommunityService Health_LifeScience': 0.5, 'CommunityService
Health_Wellness': 1.0, 'CommunityService Literacy': 1.0, 'CommunityService Literature_Writing': 0.
75, 'CommunityService Mathematics': 1.0, 'CommunityService Other': 0.0, 'CommunityService
SocialSciences': 0.5, 'CommunityService SpecialNeeds': 0.5, 'CommunityService VisualArts': 1.0, 'E
SL': 0.8360655737704918, 'ESL EarlyDevelopment': 1.0, 'ESL EnvironmentalScience':
0.6666666666666666, 'ESL Extracurricular': 1.0, 'ESL FinancialLiteracy': 0.0, 'ESL

0.6666666666666666, 'ESL Extracurricular': 1.0, 'ESL FinancialLiteracy': 0.0, 'ESL
ForeignLanguages': 0.875, 'ESL Gym_Fitness': 1.0, 'ESL Health_LifeScience': 1.0, 'ESL
Health_Wellness': 1.0, 'ESL History_Geography': 0.7142857142857143, 'ESL Literacy':
0.8842105263157894, 'ESL Literature_Writing': 0.8476190476190476, 'ESL Mathematics':
0.8888888888888888, 'ESL Music': 1.0, 'ESL Other': 1.0, 'ESL ParentInvolvement': 0.0, 'ESL
PerformingArts': 0.0, 'ESL SocialSciences': 0.5, 'ESL SpecialNeeds': 0.9629629629629629, 'ESL Visu
alArts': 1.0, 'EarlyDevelopment': 0.8245614035087719, 'EarlyDevelopment EnvironmentalScience': 0.8
333333333333334, 'EarlyDevelopment Extracurricular': 1.0, 'EarlyDevelopment Gym_Fitness': 1.0,
'EarlyDevelopment Health_LifeScience': 0.5, 'EarlyDevelopment Health_Wellness':
0.9230769230769231, 'EarlyDevelopment Literacy': 0.8837209302325582, 'EarlyDevelopment
Literature_Writing': 0.875, 'EarlyDevelopment Mathematics': 0.8536585365853658, 'EarlyDevelopment
Music': 1.0, 'EarlyDevelopment Other': 0.8, 'EarlyDevelopment ParentInvolvement':
0.8888888888888888, 'EarlyDevelopment PerformingArts': 1.0, 'EarlyDevelopment SpecialNeeds':
0.7542372881355932, 'EarlyDevelopment VisualArts': 0.7391304347826086, 'Economics': 1.0,
'Economics FinancialLiteracy': 0.7272727272727273, 'Economics ForeignLanguages': 1.0, 'Economics H
istory_Geography': 1.0, 'Economics Literacy': 1.0, 'Economics Mathematics': 1.0, 'Economics
SocialSciences': 1.0, 'Economics SpecialNeeds': 0.5, 'Economics VisualArts': 1.0,
'EnvironmentalScience': 0.8714285714285714, 'EnvironmentalScience Extracurricular': 0.0,
'EnvironmentalScience Health_LifeScience': 0.7971014492753623, 'EnvironmentalScience
Health_Wellness': 0.8181818181818182, 'EnvironmentalScience History_Geography': 1.0,
'EnvironmentalScience Literacy': 0.8533333333333334, 'EnvironmentalScience Literature_Writing': 0.
6944444444444444, 'EnvironmentalScience Mathematics': 0.7821782178217822, 'EnvironmentalScience Nu
tritionEducation': 0.8, 'EnvironmentalScience Other': 1.0, 'EnvironmentalScience
ParentInvolvement': 1.0, 'EnvironmentalScience PerformingArts': 1.0, 'EnvironmentalScience
SocialSciences': 0.6666666666666666, 'EnvironmentalScience SpecialNeeds': 0.9,
'EnvironmentalScience TeamSports': 1.0, 'EnvironmentalScience VisualArts': 0.7777777777777778, 'En
vironmentalScience Warmth Care_Hunger': 1.0, 'Extracurricular': 0.8823529411764706,
'Extracurricular Gym_Fitness': 1.0, 'Extracurricular Health_Wellness': 0.75, 'Extracurricular
Literacy': 0.8888888888888888, 'Extracurricular Literature_Writing': 0.0, 'Extracurricular
Mathematics': 0.8888888888888888, 'Extracurricular Music': 1.0, 'Extracurricular Other': 0.4, 'Ext
racurricular PerformingArts': 1.0, 'Extracurricular TeamSports': 1.0, 'Extracurricular
VisualArts': 0.8, 'FinancialLiteracy': 0.9, 'FinancialLiteracy ForeignLanguages': 1.0,
'FinancialLiteracy History_Geography': 1.0, 'FinancialLiteracy Literacy': 1.0, 'FinancialLiteracy
Literature_Writing': 1.0, 'FinancialLiteracy Mathematics': 0.9333333333333333, 'FinancialLiteracy
SpecialNeeds': 0.75, 'FinancialLiteracy VisualArts': 0.0, 'ForeignLanguages': 0.8333333333333334,
'ForeignLanguages Health_Wellness': 1.0, 'ForeignLanguages History_Geography': 0.5,
'ForeignLanguages Literacy': 0.8846153846153846, 'ForeignLanguages Literature_Writing': 0.875, 'Fo
reignLanguages Mathematics': 0.75, 'ForeignLanguages Music': 1.0, 'ForeignLanguages
SocialSciences': 0.5, 'ForeignLanguages SpecialNeeds': 1.0, 'ForeignLanguages VisualArts': 1.0, 'G
ym_Fitness': 0.8294117647058824, 'Gym_Fitness Health_Wellness': 0.8950819672131147, 'Gym_Fitness L
iteracy': 0.6, 'Gym_Fitness Literature_Writing': 0.5, 'Gym_Fitness Mathematics': 0.6, 'Gym_Fitness
Music': 1.0, 'Gym_Fitness NutritionEducation': 0.75, 'Gym_Fitness ParentInvolvement': 1.0,
'Gym_Fitness PerformingArts': 1.0, 'Gym_Fitness SpecialNeeds': 0.85, 'Gym_Fitness TeamSports': 0.7
5, 'Gym_Fitness VisualArts': 0.8, 'Health_LifeScience': 0.8017241379310345, 'Health_LifeScience He
alth_Wellness': 0.9444444444444444, 'Health_LifeScience History_Geography': 0.9230769230769231, 'H
ealth_LifeScience Literacy': 0.9111111111111111, 'Health_LifeScience Literature_Writing':
0.9285714285714286, 'Health_LifeScience Mathematics': 0.7313432835820896, 'Health_LifeScience
Music': 1.0, 'Health_LifeScience NutritionEducation': 0.5, 'Health_LifeScience Other': 1.0,
'Health_LifeScience ParentInvolvement': 1.0, 'Health_LifeScience PerformingArts':
0.6666666666666666, 'Health_LifeScience SocialSciences': 1.0, 'Health_LifeScience SpecialNeeds': 0
.7, 'Health_LifeScience TeamSports': 1.0, 'Health_LifeScience VisualArts': 0.7058823529411765, 'He
alth_Wellness': 0.878727634194831, 'Health_Wellness History_Geography': 0.8, 'Health_Wellness Lite
racy': 0.9344262295081968, 'Health_Wellness Literature_Writing': 0.9090909090909091,
'Health_Wellness Mathematics': 0.8620689655172413, 'Health_Wellness Music': 0.6666666666666666, 'H
ealth_Wellness NutritionEducation': 0.8557692307692307, 'Health_Wellness Other': 0.9,
'Health_Wellness PerformingArts': 1.0, 'Health_Wellness SocialSciences': 1.0, 'Health_Wellness Spe
cialNeeds': 0.8940397350993378, 'Health_Wellness TeamSports': 0.8163265306122449, 'Health_Wellness
VisualArts': 1.0, 'Health_Wellness Warmth Care_Hunger': 1.0, 'History_Geography': 0.825,
'History_Geography Literacy': 0.9264705882352942, 'History_Geography Literature_Writing': 0.875, '
History_Geography Mathematics': 0.7777777777777778, 'History_Geography Music': 1.0,
'History_Geography ParentInvolvement': 1.0, 'History_Geography PerformingArts':
0.6666666666666666, 'History_Geography SocialSciences': 0.7906976744186046, 'History_Geography
SpecialNeeds': 0.8571428571428571, 'History_Geography TeamSports': 1.0, 'History_Geography
VisualArts': 0.7727272727272727, 'Literacy': 0.8803611738148984, 'Literacy Literature_Writing': 0.
861323155216285, 'Literacy Mathematics': 0.8792650918635171, 'Literacy Music': 0.8928571428571429,
'Literacy Other': 0.84, 'Literacy ParentInvolvement': 0.9523809523809523, 'Literacy
PerformingArts': 0.7777777777777778, 'Literacy SocialSciences': 0.8888888888888888, 'Literacy Spec
ialNeeds': 0.8579545454545454, 'Literacy VisualArts': 0.8354430379746836, 'Literacy Warmth
Care_Hunger': 1.0, 'Literature_Writing': 0.838351822503962, 'Literature_Writing Mathematics':
0.8745247148288974, 'Literature_Writing Music': 1.0, 'Literature_Writing Other': 0.6875,
'Literature_Writing ParentInvolvement': 1.0, 'Literature_Writing PerformingArts':
0.9285714285714286, 'Literature_Writing SocialSciences': 0.95, 'Literature_Writing SpecialNeeds':
0.8430232558139535, 'Literature_Writing VisualArts': 0.7972972972972973, 'Mathematics':
0.8144736842105263, 'Mathematics Music': 1.0, 'Mathematics NutritionEducation': 1.0, 'Mathematics
Other': 0.7777777777777778, 'Mathematics ParentInvolvement': 0.9285714285714286, 'Mathematics Perf
ormingArts': 1.0, 'Mathematics SocialSciences': 0.9, 'Mathematics SpecialNeeds':
0.7933333333333333, 'Mathematics VisualArts': 0.7931034482758621, 'Music': 0.8867924528301887, 'Mu
sic Other': 0.0, 'Music ParentInvolvement': 0.0, 'Music PerformingArts': 0.916030534351145, 'Music

sic Other': 0.0, 'Music ParentInvolvement': 0.0, 'Music PerformingArts': 0.916030534351145, 'Music
SocialSciences': 1.0, 'Music SpecialNeeds': 1.0, 'Music TeamSports': 0.5, 'Music VisualArts': 1.0,
'NutritionEducation': 0.8333333333333334, 'NutritionEducation Other': 1.0, 'NutritionEducation Spe
cialNeeds': 1.0, 'NutritionEducation TeamSports': 0.6666666666666666, 'NutritionEducation
VisualArts': 0.0, 'Other': 0.8558558558558559, 'Other ParentInvolvement': 1.0, 'Other
SocialSciences': 1.0, 'Other SpecialNeeds': 0.8076923076923077, 'Other VisualArts':
0.8571428571428571, 'ParentInvolvement': 0.75, 'ParentInvolvement VisualArts': 1.0,
'PerformingArts': 0.8823529411764706, 'PerformingArts SpecialNeeds': 1.0, 'PerformingArts
TeamSports': 1.0, 'PerformingArts VisualArts': 0.8461538461538461, 'SocialSciences': 1.0,
'SocialSciences SpecialNeeds': 0.6, 'SocialSciences TeamSports': 1.0, 'SocialSciences VisualArts':
0.8, 'SpecialNeeds': 0.8249594813614263, 'SpecialNeeds TeamSports': 1.0, 'SpecialNeeds
VisualArts': 0.8367346938775511, 'SpecialNeeds Warmth Care_Hunger': 0.8, 'TeamSports':
0.8333333333333334, 'VisualArts': 0.8381877022653722, 'Warmth Care_Hunger': 0.9326424870466321}
{'AppliedSciences': 0.13934426229508196, 'AppliedSciences CharacterEducation': 0.0,
'AppliedSciences Civics_Government': 0.0, 'AppliedSciences College_CareerPrep':
0.16666666666666666, 'AppliedSciences CommunityService': 0.0, 'AppliedSciences ESL': 0.0,
'AppliedSciences EarlyDevelopment': 0.0, 'AppliedSciences EnvironmentalScience':
0.13333333333333333, 'AppliedSciences Extracurricular': 0.0, 'AppliedSciences FinancialLiteracy':
0.0, 'AppliedSciences Gym_Fitness': 0.0, 'AppliedSciences Health_LifeScience':
0.16071428571428573, 'AppliedSciences Health_Wellness': 0.0, 'AppliedSciences History_Geography':
0.2, 'AppliedSciences Literacy': 0.1774193548387097, 'AppliedSciences Literature_Writing': 0.15, '
AppliedSciences Mathematics': 0.18100890207715134, 'AppliedSciences Music': 0.0, 'AppliedSciences
Other': 0.06666666666666667, 'AppliedSciences ParentInvolvement': 0.0, 'AppliedSciences
PerformingArts': 0.0, 'AppliedSciences SocialSciences': 0.2, 'AppliedSciences SpecialNeeds': 0.161
29032258064516, 'AppliedSciences TeamSports': 0.0, 'AppliedSciences VisualArts':
0.21212121212121213, 'CharacterEducation': 0.07142857142857142, 'CharacterEducation
College_CareerPrep': 0.2857142857142857, 'CharacterEducation CommunityService':
0.3333333333333333, 'CharacterEducation ESL': 0.0, 'CharacterEducation EarlyDevelopment':
0.14285714285714285, 'CharacterEducation Economics': 1.0, 'CharacterEducation
EnvironmentalScience': 0.3333333333333333, 'CharacterEducation Extracurricular': 0.5,
'CharacterEducation ForeignLanguages': 0.0, 'CharacterEducation Health_LifeScience': 1.0,
'CharacterEducation Health_Wellness': 0.3333333333333333, 'CharacterEducation History_Geography':
0.0, 'CharacterEducation Literacy': 0.12121212121212122, 'CharacterEducation Literature_Writing':
0.19047619047619047, 'CharacterEducation Mathematics': 0.10526315789473684, 'CharacterEducation Mu
sic': 0.0, 'CharacterEducation Other': 0.25, 'CharacterEducation ParentInvolvement': 0.4,
'CharacterEducation PerformingArts': 1.0, 'CharacterEducation SocialSciences': 0.0,
'CharacterEducation SpecialNeeds': 0.2222222222222222, 'CharacterEducation TeamSports':
0.6666666666666666, 'CharacterEducation VisualArts': 0.2727272727272727, 'CharacterEducation
Warmth Care_Hunger': 0.0, 'Civics_Government': 0.125, 'Civics_Government CommunityService': 0.0,
'Civics_Government Economics': 0.0, 'Civics_Government Extracurricular': 0.0, 'Civics_Government F
inancialLiteracy': 0.0, 'Civics_Government Health_LifeScience': 0.0, 'Civics_Government
Health_Wellness': 0.0, 'Civics_Government History_Geography': 0.15384615384615385,
'Civics_Government Literacy': 0.125, 'Civics_Government Literature_Writing': 0.18181818181818182,
'Civics_Government Mathematics': 1.0, 'Civics_Government PerformingArts': 0.0, 'Civics_Government
SocialSciences': 0.0, 'Civics_Government SpecialNeeds': 0.0, 'Civics_Government VisualArts': 0.0,
'College_CareerPrep': 0.22727272727272727, 'College_CareerPrep CommunityService': 0.0,
'College_CareerPrep ESL': 0.0, 'College_CareerPrep EarlyDevelopment': 0.0, 'College_CareerPrep
Economics': 0.0, 'College_CareerPrep EnvironmentalScience': 0.0, 'College_CareerPrep
Extracurricular': 0.0, 'College_CareerPrep FinancialLiteracy': 1.0, 'College_CareerPrep
ForeignLanguages': 0.0, 'College_CareerPrep Health_LifeScience': 0.0, 'College_CareerPrep
Health_Wellness': 0.0, 'College_CareerPrep History_Geography': 1.0, 'College_CareerPrep Literacy':
0.09523809523809523, 'College_CareerPrep Literature_Writing': 0.16666666666666666,
'College_CareerPrep Mathematics': 0.13793103448275862, 'College_CareerPrep NutritionEducation':
0.3333333333333333, 'College_CareerPrep Other': 0.2727272727272727, 'College_CareerPrep
ParentInvolvement': 0.0, 'College_CareerPrep PerformingArts': 0.0, 'College_CareerPrep
SocialSciences': 0.25, 'College_CareerPrep SpecialNeeds': 0.4375, 'College_CareerPrep VisualArts':
0.0, 'CommunityService': 0.0, 'CommunityService EnvironmentalScience': 0.25, 'CommunityService Ext
racurricular': 0.0, 'CommunityService Health_LifeScience': 0.0, 'CommunityService
Health_Wellness': 0.0, 'CommunityService History_Geography': 0.0, 'CommunityService Literacy': 0.5
, 'CommunityService Literature_Writing': 0.5, 'CommunityService Mathematics': 1.0,
'CommunityService Other': 1.0, 'CommunityService ParentInvolvement': 0.0, 'CommunityService
SpecialNeeds': 0.3333333333333333, 'CommunityService VisualArts': 0.5, 'ESL': 0.10416666666666667,
'ESL EarlyDevelopment': 0.0, 'ESL EnvironmentalScience': 0.25, 'ESL ForeignLanguages': 0.2, 'ESL H
ealth_LifeScience': 0.0, 'ESL Health_Wellness': 0.0, 'ESL History_Geography': 0.0, 'ESL Literacy':
0.125, 'ESL Literature_Writing': 0.13253012048192772, 'ESL Mathematics': 0.06451612903225806, 'ESL
Music': 0.0, 'ESL NutritionEducation': 0.0, 'ESL Other': 0.0, 'ESL ParentInvolvement': 0.0, 'ESL P
erformingArts': 0.0, 'ESL SocialSciences': 0.0, 'ESL SpecialNeeds': 0.13333333333333333, 'ESL Visu
alArts': 0.2, 'EarlyDevelopment': 0.13978494623655913, 'EarlyDevelopment Economics': 0.0,
'EarlyDevelopment EnvironmentalScience': 0.0, 'EarlyDevelopment Health_LifeScience': 0.0,
'EarlyDevelopment Health_Wellness': 0.16129032258064516, 'EarlyDevelopment History_Geography': 0.0
, 'EarlyDevelopment Literacy': 0.19642857142857142, 'EarlyDevelopment Literature_Writing':
0.17647058823529413, 'EarlyDevelopment Mathematics': 0.2, 'EarlyDevelopment Music': 0.25,
'EarlyDevelopment Other': 0.17647058823529413, 'EarlyDevelopment ParentInvolvement': 0.0,
'EarlyDevelopment PerformingArts': 0.0, 'EarlyDevelopment SocialSciences': 0.0, 'EarlyDevelopment
SpecialNeeds': 0.13043478260869565, 'EarlyDevelopment TeamSports': 1.0, 'EarlyDevelopment
VisualArts': 0.2, 'EarlyDevelopment Warmth Care_Hunger': 0.0, 'Economics': 0.0, 'Economics
FinancialLiteracy': 0.14285714285714285, 'Economics History_Geography': 0.0, 'Economics Literacy':
0.0, 'Economics Mathematics': 0.0, 'Economics VisualArts': 0.0, 'EnvironmentalScience':

0.0, 'Economics Mathematics': 0.0, 'Economics VisualArts': 0.0, 'EnvironmentalScience':
0.12727272727272726, 'EnvironmentalScience Extracurricular': 0.0, 'EnvironmentalScience
FinancialLiteracy': 0.0, 'EnvironmentalScience Health_LifeScience': 0.13592233009708737,
'EnvironmentalScience Health_Wellness': 0.3333333333333333, 'EnvironmentalScience
History_Geography': 0.17647058823529413, 'EnvironmentalScience Literacy': 0.11904761904761904,
'EnvironmentalScience Literature_Writing': 0.25, 'EnvironmentalScience Mathematics':
0.18888888888888888, 'EnvironmentalScience Music': 1.0, 'EnvironmentalScience NutritionEducation':
0.0, 'EnvironmentalScience Other': 0.0, 'EnvironmentalScience SocialSciences': 0.2,
'EnvironmentalScience SpecialNeeds': 0.21428571428571427, 'EnvironmentalScience VisualArts': 0.25,
'EnvironmentalScience Warmth Care_Hunger': 1.0, 'Extracurricular': 0.21428571428571427,
'Extracurricular Gym_Fitness': 0.0, 'Extracurricular Health_LifeScience': 0.0, 'Extracurricular Li
teracy': 0.25, 'Extracurricular Literature_Writing': 0.0, 'Extracurricular Mathematics': 0.0,
'Extracurricular Music': 0.0, 'Extracurricular Other': 0.0, 'Extracurricular ParentInvolvement': 0
.0, 'Extracurricular PerformingArts': 0.0, 'Extracurricular TeamSports': 0.0, 'Extracurricular Vis
ualArts': 0.18181818181818182, 'FinancialLiteracy': 0.3333333333333333, 'FinancialLiteracy
Health_Wellness': 0.0, 'FinancialLiteracy History_Geography': 0.0, 'FinancialLiteracy Literacy': 0
.0, 'FinancialLiteracy Literature_Writing': 0.0, 'FinancialLiteracy Mathematics':
0.14285714285714285, 'FinancialLiteracy ParentInvolvement': 0.0, 'FinancialLiteracy
PerformingArts': 0.0, 'FinancialLiteracy SpecialNeeds': 0.25, 'FinancialLiteracy VisualArts': 0.0,
'ForeignLanguages': 0.26666666666666666, 'ForeignLanguages Health_Wellness': 0.0,
'ForeignLanguages Literacy': 0.28, 'ForeignLanguages Literature_Writing': 0.1111111111111111,
'ForeignLanguages Mathematics': 0.5, 'ForeignLanguages Music': 1.0, 'ForeignLanguages Other': 0.0,
'ForeignLanguages PerformingArts': 0.0, 'ForeignLanguages VisualArts': 1.0, 'Gym_Fitness':
0.1056910569105691, 'Gym_Fitness Health_LifeScience': 0.0, 'Gym_Fitness Health_Wellness':
0.14847161572052403, 'Gym_Fitness Literacy': 0.5, 'Gym_Fitness Literature_Writing': 0.0,
'Gym_Fitness Mathematics': 0.0, 'Gym_Fitness Music': 0.3333333333333333, 'Gym_Fitness
NutritionEducation': 0.4, 'Gym_Fitness PerformingArts': 0.5, 'Gym_Fitness SocialSciences': 0.0, 'G
ym_Fitness SpecialNeeds': 0.18181818181818182, 'Gym_Fitness TeamSports': 0.19607843137254902,
'Gym_Fitness VisualArts': 0.5, 'Health_LifeScience': 0.12658227848101267, 'Health_LifeScience Heal
th_Wellness': 0.22727272727272727, 'Health_LifeScience History_Geography': 1.0,
'Health_LifeScience Literacy': 0.0625, 'Health_LifeScience Literature_Writing':
0.07142857142857142, 'Health_LifeScience Mathematics': 0.17391304347826086, 'Health_LifeScience Nu
tritionEducation': 0.25, 'Health_LifeScience Other': 0.0, 'Health_LifeScience ParentInvolvement':
0.0, 'Health_LifeScience SocialSciences': 0.16666666666666666, 'Health_LifeScience SpecialNeeds':
0.0, 'Health_LifeScience TeamSports': 0.0, 'Health_LifeScience VisualArts': 0.0,
'Health_LifeScience Warmth Care_Hunger': 0.0, 'Health_Wellness': 0.15479876160990713,
'Health_Wellness History_Geography': 0.0, 'Health_Wellness Literacy': 0.1836734693877551,
'Health_Wellness Literature_Writing': 0.05263157894736842, 'Health_Wellness Mathematics':
0.08695652173913043, 'Health_Wellness Music': 0.0, 'Health_Wellness NutritionEducation': 0.2, 'Hea
lth_Wellness Other': 0.0, 'Health_Wellness SocialSciences': 0.0, 'Health_Wellness SpecialNeeds': 0
.12295081967213115, 'Health_Wellness TeamSports': 0.25, 'Health_Wellness VisualArts':
0.16666666666666666, 'Health_Wellness Warmth Care_Hunger': 0.0, 'History_Geography':
0.18181818181818182, 'History_Geography Literacy': 0.04081632653061224, 'History_Geography
Literature_Writing': 0.13636363636363635, 'History_Geography Mathematics': 0.0, 'History_Geography
Music': 0.3333333333333333, 'History_Geography SocialSciences': 0.1111111111111111,
'History_Geography SpecialNeeds': 0.07692307692307693, 'History_Geography VisualArts': 0.15,
'History_Geography Warmth Care_Hunger': 1.0, 'Literacy': 0.10835214446952596, 'Literacy
Literature_Writing': 0.14531548757170173, 'Literacy Mathematics': 0.11207970112079702, 'Literacy M
usic': 0.0, 'Literacy NutritionEducation': 0.0, 'Literacy Other': 0.2, 'Literacy
ParentInvolvement': 0.0625, 'Literacy PerformingArts': 0.0, 'Literacy SocialSciences':
0.12244897959183673, 'Literacy SpecialNeeds': 0.13679245283018868, 'Literacy TeamSports': 0.0, 'Li
teracy VisualArts': 0.2, 'Literacy Warmth Care_Hunger': 0.5, 'Literature_Writing':
0.15609756097560976, 'Literature_Writing Mathematics': 0.14010507880910683, 'Literature_Writing Mu
sic': 0.2, 'Literature_Writing Other': 0.25, 'Literature_Writing ParentInvolvement': 0.0,
'Literature_Writing PerformingArts': 0.0, 'Literature_Writing SocialSciences':
0.23684210526315788, 'Literature_Writing SpecialNeeds': 0.11678832116788321, 'Literature_Writing T
eamSports': 0.0, 'Literature_Writing VisualArts': 0.14285714285714285, 'Literature_Writing Warmth
Care_Hunger': 0.0, 'Mathematics': 0.20555555555555555, 'Mathematics Music': 0.3333333333333333, 'M
athematics Other': 0.0, 'Mathematics ParentInvolvement': 0.08333333333333333, 'Mathematics
PerformingArts': 0.0, 'Mathematics SocialSciences': 0.1111111111111111, 'Mathematics
SpecialNeeds': 0.22641509433962265, 'Mathematics VisualArts': 0.18181818181818182, 'Mathematics
Warmth Care_Hunger': 1.0, 'Music': 0.11678832116788321, 'Music ParentInvolvement': 0.0, 'Music Per
formingArts': 0.13157894736842105, 'Music SpecialNeeds': 0.17647058823529413, 'Music TeamSports':
1.0, 'Music VisualArts': 0.16666666666666666, 'NutritionEducation': 0.22857142857142856,
'NutritionEducation Other': 0.0, 'NutritionEducation SpecialNeeds': 0.25, 'NutritionEducation
TeamSports': 0.5, 'NutritionEducation Warmth Care_Hunger': 0.0, 'Other': 0.08620689655172414, 'Oth
er SpecialNeeds': 0.2, 'Other VisualArts': 0.16666666666666666, 'ParentInvolvement': 0.6,
'ParentInvolvement SpecialNeeds': 0.0, 'ParentInvolvement VisualArts': 0.0, 'PerformingArts':
0.11538461538461539, 'PerformingArts SocialSciences': 1.0, 'PerformingArts SpecialNeeds': 0.0, 'Pe
rformingArts VisualArts': 0.5, 'SocialSciences': 0.19230769230769232, 'SocialSciences
SpecialNeeds': 0.25, 'SocialSciences VisualArts': 0.0, 'SpecialNeeds': 0.22588235294117648,
'SpecialNeeds TeamSports': 0.25, 'SpecialNeeds VisualArts': 0.18181818181818182, 'SpecialNeeds
Warmth Care_Hunger': 0.5, 'TeamSports': 0.2018348623853211, 'TeamSports VisualArts': 0.5,
'VisualArts': 0.1746031746031746, 'Warmth Care_Hunger': 0.08571428571428572}
{'AppliedSciences': 0.860655737704918, 'AppliedSciences CharacterEducation': 1.0, 'AppliedSciences
Civics_Government': 1.0, 'AppliedSciences College_CareerPrep': 0.8333333333333334,
'AppliedSciences CommunityService': 1.0, 'AppliedSciences ESL': 1.0, 'AppliedSciences

EarlyDevelopment': 1.0, 'AppliedSciences EnvironmentalScience': 0.8666666666666667,
'AppliedSciences Extracurricular': 1.0, 'AppliedSciences FinancialLiteracy': 1.0, 'AppliedSciences
Gym_Fitness': 1.0, 'AppliedSciences Health_LifeScience': 0.8392857142857143, 'AppliedSciences Heal
th_Wellness': 1.0, 'AppliedSciences History_Geography': 0.8, 'AppliedSciences Literacy':
0.8225806451612904, 'AppliedSciences Literature_Writing': 0.85, 'AppliedSciences Mathematics':
0.8189910979228486, 'AppliedSciences Music': 1.0, 'AppliedSciences Other': 0.9333333333333333, 'Ap
pliedSciences ParentInvolvement': 1.0, 'AppliedSciences PerformingArts': 1.0, 'AppliedSciences Soc
ialSciences': 0.8, 'AppliedSciences SpecialNeeds': 0.8387096774193549, 'AppliedSciences
TeamSports': 1.0, 'AppliedSciences VisualArts': 0.7878787878787878, 'CharacterEducation':
0.9285714285714286, 'CharacterEducation College_CareerPrep': 0.7142857142857143,
'CharacterEducation CommunityService': 0.6666666666666666, 'CharacterEducation ESL': 1.0,
'CharacterEducation EarlyDevelopment': 0.8571428571428571, 'CharacterEducation Economics': 0.0, 'C
haracterEducation EnvironmentalScience': 0.6666666666666666, 'CharacterEducation Extracurricular':
0.5, 'CharacterEducation ForeignLanguages': 1.0, 'CharacterEducation Health_LifeScience': 0.0, 'Ch
aracterEducation Health_Wellness': 0.6666666666666666, 'CharacterEducation History_Geography': 1.0
, 'CharacterEducation Literacy': 0.8787878787878788, 'CharacterEducation Literature_Writing':
0.8095238095238095, 'CharacterEducation Mathematics': 0.8947368421052632, 'CharacterEducation
Music': 1.0, 'CharacterEducation Other': 0.75, 'CharacterEducation ParentInvolvement': 0.6,
'CharacterEducation PerformingArts': 0.0, 'CharacterEducation SocialSciences': 1.0,
'CharacterEducation SpecialNeeds': 0.7777777777777778, 'CharacterEducation TeamSports':
0.3333333333333333, 'CharacterEducation VisualArts': 0.7272727272727273, 'CharacterEducation
Warmth Care_Hunger': 1.0, 'Civics_Government': 0.875, 'Civics_Government CommunityService': 1.0,
'Civics_Government Economics': 1.0, 'Civics_Government Extracurricular': 1.0, 'Civics_Government F
inancialLiteracy': 1.0, 'Civics_Government Health_LifeScience': 1.0, 'Civics_Government
Health_Wellness': 1.0, 'Civics_Government History_Geography': 0.8461538461538461,
'Civics_Government Literacy': 0.875, 'Civics_Government Literature_Writing': 0.8181818181818182, '
Civics_Government Mathematics': 0.0, 'Civics_Government PerformingArts': 1.0, 'Civics_Government S
ocialSciences': 1.0, 'Civics_Government SpecialNeeds': 1.0, 'Civics_Government VisualArts': 1.0, '
College_CareerPrep': 0.7727272727272727, 'College_CareerPrep CommunityService': 1.0,
'College_CareerPrep ESL': 1.0, 'College_CareerPrep EarlyDevelopment': 1.0, 'College_CareerPrep
Economics': 1.0, 'College_CareerPrep EnvironmentalScience': 1.0, 'College_CareerPrep
Extracurricular': 1.0, 'College_CareerPrep FinancialLiteracy': 0.0, 'College_CareerPrep
ForeignLanguages': 1.0, 'College_CareerPrep Health_LifeScience': 1.0, 'College_CareerPrep
Health_Wellness': 1.0, 'College_CareerPrep History_Geography': 0.0, 'College_CareerPrep Literacy':
0.9047619047619048, 'College_CareerPrep Literature_Writing': 0.8333333333333334,
'College_CareerPrep Mathematics': 0.8620689655172413, 'College_CareerPrep NutritionEducation':
0.6666666666666666, 'College_CareerPrep Other': 0.7272727272727273, 'College_CareerPrep
ParentInvolvement': 1.0, 'College_CareerPrep PerformingArts': 1.0, 'College_CareerPrep
SocialSciences': 0.75, 'College_CareerPrep SpecialNeeds': 0.5625, 'College_CareerPrep VisualArts':
1.0, 'CommunityService': 1.0, 'CommunityService EnvironmentalScience': 0.75, 'CommunityService Ext
racurricular': 1.0, 'CommunityService Health_LifeScience': 1.0, 'CommunityService
Health_Wellness': 1.0, 'CommunityService History_Geography': 1.0, 'CommunityService Literacy': 0.5
, 'CommunityService Literature_Writing': 0.5, 'CommunityService Mathematics': 0.0,
'CommunityService Other': 0.0, 'CommunityService ParentInvolvement': 1.0, 'CommunityService
SpecialNeeds': 0.6666666666666666, 'CommunityService VisualArts': 0.5, 'ESL': 0.8958333333333334,
'ESL EarlyDevelopment': 1.0, 'ESL EnvironmentalScience': 0.75, 'ESL ForeignLanguages': 0.8, 'ESL H
ealth_LifeScience': 1.0, 'ESL Health_Wellness': 1.0, 'ESL History_Geography': 1.0, 'ESL Literacy':
0.875, 'ESL Literature_Writing': 0.8674698795180723, 'ESL Mathematics': 0.9354838709677419, 'ESL M
usic': 1.0, 'ESL NutritionEducation': 1.0, 'ESL Other': 1.0, 'ESL ParentInvolvement': 1.0, 'ESL Pe
rformingArts': 1.0, 'ESL SocialSciences': 1.0, 'ESL SpecialNeeds': 0.8666666666666667, 'ESL
VisualArts': 0.8, 'EarlyDevelopment': 0.8602150537634409, 'EarlyDevelopment Economics': 1.0, 'Earl
yDevelopment EnvironmentalScience': 1.0, 'EarlyDevelopment Health_LifeScience': 1.0,
'EarlyDevelopment Health_Wellness': 0.8387096774193549, 'EarlyDevelopment History_Geography': 1.0,
'EarlyDevelopment Literacy': 0.8035714285714286, 'EarlyDevelopment Literature_Writing':
0.8235294117647058, 'EarlyDevelopment Mathematics': 0.8, 'EarlyDevelopment Music': 0.75,
'EarlyDevelopment Other': 0.8235294117647058, 'EarlyDevelopment ParentInvolvement': 1.0,
'EarlyDevelopment PerformingArts': 1.0, 'EarlyDevelopment SocialSciences': 1.0, 'EarlyDevelopment
SpecialNeeds': 0.8695652173913043, 'EarlyDevelopment TeamSports': 0.0, 'EarlyDevelopment
VisualArts': 0.8, 'EarlyDevelopment Warmth Care_Hunger': 1.0, 'Economics': 1.0, 'Economics
FinancialLiteracy': 0.8571428571428571, 'Economics History_Geography': 1.0, 'Economics Literacy':
1.0, 'Economics Mathematics': 1.0, 'Economics VisualArts': 1.0, 'EnvironmentalScience':
0.8727272727272727, 'EnvironmentalScience Extracurricular': 1.0, 'EnvironmentalScience
FinancialLiteracy': 1.0, 'EnvironmentalScience Health_LifeScience': 0.8640776699029126,
'EnvironmentalScience Health_Wellness': 0.6666666666666666, 'EnvironmentalScience
History_Geography': 0.8235294117647058, 'EnvironmentalScience Literacy': 0.8809523809523809,
'EnvironmentalScience Literature_Writing': 0.75, 'EnvironmentalScience Mathematics':
0.8111111111111111, 'EnvironmentalScience Music': 0.0, 'EnvironmentalScience NutritionEducation':
1.0, 'EnvironmentalScience Other': 1.0, 'EnvironmentalScience SocialSciences': 0.8,
'EnvironmentalScience SpecialNeeds': 0.7857142857142857, 'EnvironmentalScience VisualArts': 0.75,
'EnvironmentalScience Warmth Care_Hunger': 0.0, 'Extracurricular': 0.7857142857142857,
'Extracurricular Gym_Fitness': 1.0, 'Extracurricular Health_LifeScience': 1.0, 'Extracurricular Li
teracy': 0.75, 'Extracurricular Literature_Writing': 1.0, 'Extracurricular Mathematics': 1.0,
'Extracurricular Music': 1.0, 'Extracurricular Other': 1.0, 'Extracurricular ParentInvolvement': 1
.0, 'Extracurricular PerformingArts': 1.0, 'Extracurricular TeamSports': 1.0, 'Extracurricular Vis
ualArts': 0.8181818181818182, 'FinancialLiteracy': 0.6666666666666666, 'FinancialLiteracy
Health_Wellness': 1.0, 'FinancialLiteracy History_Geography': 1.0, 'FinancialLiteracy Literacy': 1
.0, 'FinancialLiteracy Literature_Writing': 1.0, 'FinancialLiteracy Mathematics':

```
0.8571428571428571, 'FinancialLiteracy ParentInvolvement': 1.0, 'FinancialLiteracy
PerformingArts': 1.0, 'FinancialLiteracy SpecialNeeds': 0.75, 'FinancialLiteracy VisualArts': 1.0,
'ForeignLanguages': 0.7333333333333333, 'ForeignLanguages Health_Wellness': 1.0, 'ForeignLanguages
Literacy': 0.72, 'ForeignLanguages Literature_Writing': 0.8888888888888888, 'ForeignLanguages
Mathematics': 0.5, 'ForeignLanguages Music': 0.0, 'ForeignLanguages Other': 1.0, 'ForeignLanguages
PerformingArts': 1.0, 'ForeignLanguages VisualArts': 0.0, 'Gym_Fitness': 0.8943089430894309,
'Gym_Fitness Health_LifeScience': 1.0, 'Gym_Fitness Health_Wellness': 0.851528384279476,
'Gym_Fitness Literacy': 0.5, 'Gym_Fitness Literature_Writing': 1.0, 'Gym_Fitness Mathematics':
1.0, 'Gym_Fitness Music': 0.6666666666666666, 'Gym_Fitness NutritionEducation': 0.6, 'Gym_Fitness
PerformingArts': 0.5, 'Gym_Fitness SocialSciences': 1.0, 'Gym_Fitness SpecialNeeds':
0.8181818181818182, 'Gym_Fitness TeamSports': 0.803921568627451, 'Gym_Fitness VisualArts': 0.5,
'Health_LifeScience': 0.8734177215189873, 'Health_LifeScience Health_Wellness':
0.7727272727272727, 'Health_LifeScience History_Geography': 0.0, 'Health_LifeScience Literacy': 0.
9375, 'Health_LifeScience Literature_Writing': 0.9285714285714286, 'Health_LifeScience
Mathematics': 0.8260869565217391, 'Health_LifeScience NutritionEducation': 0.75,
'Health_LifeScience Other': 1.0, 'Health_LifeScience ParentInvolvement': 1.0, 'Health_LifeScience
SocialSciences': 0.8333333333333334, 'Health_LifeScience SpecialNeeds': 1.0, 'Health_LifeScience T
eamSports': 1.0, 'Health_LifeScience VisualArts': 1.0, 'Health_LifeScience Warmth Care_Hunger': 1.
0, 'Health_Wellness': 0.8452012383900929, 'Health_Wellness History_Geography': 1.0,
'Health_Wellness Literacy': 0.8163265306122449, 'Health_Wellness Literature_Writing':
0.9473684210526315, 'Health_Wellness Mathematics': 0.9130434782608695, 'Health_Wellness Music': 1.
0, 'Health_Wellness NutritionEducation': 0.8, 'Health_Wellness Other': 1.0, 'Health_Wellness Socia
lSciences': 1.0, 'Health_Wellness SpecialNeeds': 0.8770491803278688, 'Health_Wellness TeamSports':
0.75, 'Health_Wellness VisualArts': 0.8333333333333334, 'Health_Wellness Warmth Care_Hunger': 1.0,
'History_Geography': 0.8181818181818182, 'History_Geography Literacy': 0.9591836734693877,
'History_Geography Literature_Writing': 0.8636363636363636, 'History_Geography Mathematics': 1.0,
'History_Geography Music': 0.6666666666666666, 'History_Geography SocialSciences':
0.8888888888888888, 'History_Geography SpecialNeeds': 0.9230769230769231, 'History_Geography
VisualArts': 0.85, 'History_Geography Warmth Care_Hunger': 0.0, 'Literacy': 0.891647855530474,
'Literacy Literature_Writing': 0.8546845124282982, 'Literacy Mathematics': 0.887920298879203,
'Literacy Music': 1.0, 'Literacy NutritionEducation': 1.0, 'Literacy Other': 0.8, 'Literacy
ParentInvolvement': 0.9375, 'Literacy PerformingArts': 1.0, 'Literacy SocialSciences':
0.8775510204081632, 'Literacy SpecialNeeds': 0.8632075471698113, 'Literacy TeamSports': 1.0,
'Literacy VisualArts': 0.8, 'Literacy Warmth Care_Hunger': 0.5, 'Literature_Writing':
0.8439024390243902, 'Literature_Writing Mathematics': 0.8598949211908932, 'Literature_Writing
Music': 0.8, 'Literature_Writing Other': 0.75, 'Literature_Writing ParentInvolvement': 1.0,
'Literature_Writing PerformingArts': 1.0, 'Literature_Writing SocialSciences': 0.7631578947368421,
'Literature_Writing SpecialNeeds': 0.8832116788321168, 'Literature_Writing TeamSports': 1.0,
'Literature_Writing VisualArts': 0.8571428571428571, 'Literature_Writing Warmth Care_Hunger': 1.0,
'Mathematics': 0.7944444444444444, 'Mathematics Music': 0.6666666666666666, 'Mathematics Other': 1
.0, 'Mathematics ParentInvolvement': 0.9166666666666666, 'Mathematics PerformingArts': 1.0,
'Mathematics SocialSciences': 0.8888888888888888, 'Mathematics SpecialNeeds': 0.7735849056603774,
'Mathematics VisualArts': 0.8181818181818182, 'Mathematics Warmth Care_Hunger': 0.0, 'Music': 0.88
32116788321168, 'Music ParentInvolvement': 1.0, 'Music PerformingArts': 0.868421052631579, 'Music
SpecialNeeds': 0.8235294117647058, 'Music TeamSports': 0.0, 'Music VisualArts':
0.8333333333333334, 'NutritionEducation': 0.7714285714285715, 'NutritionEducation Other': 1.0, 'Nu
tritionEducation SpecialNeeds': 0.75, 'NutritionEducation TeamSports': 0.5, 'NutritionEducation Wa
rmth Care_Hunger': 1.0, 'Other': 0.9137931034482759, 'Other SpecialNeeds': 0.8, 'Other
VisualArts': 0.8333333333333334, 'ParentInvolvement': 0.4, 'ParentInvolvement SpecialNeeds': 1.0,
'ParentInvolvement VisualArts': 1.0, 'PerformingArts': 0.8846153846153846, 'PerformingArts
SocialSciences': 0.0, 'PerformingArts SpecialNeeds': 1.0, 'PerformingArts VisualArts': 0.5,
'SocialSciences': 0.8076923076923077, 'SocialSciences SpecialNeeds': 0.75, 'SocialSciences
VisualArts': 1.0, 'SpecialNeeds': 0.7741176470588236, 'SpecialNeeds TeamSports': 0.75,
'SpecialNeeds VisualArts': 0.8181818181818182, 'SpecialNeeds Warmth Care_Hunger': 0.5,
'TeamSports': 0.7981651376146789, 'TeamSports VisualArts': 0.5, 'VisualArts': 0.8253968253968254,
'Warmth Care_Hunger': 0.9142857142857143}
```


Vectorize the Categorical Features - school state

In [33]:

```
#School State
state_0_train = get_response(X_train['school_state'],y_train)[0]
state_1_train = get_response(X_train['school_state'],y_train)[1]

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

state_0_cv = get_response(X_cv['school_state'],y_cv)[0]
state_1_cv = get_response(X_cv['school_state'],y_cv)[1]

print(state_0_train)
print(state_1_train)

print(state_0_test)
```

```
print(state_1_test)

print(state_0_cv)
print(state_1_cv)

#==============================add to train df
state_neg_train = []
state_pos_train = []
for i in X_train['school_state']:
    state_neg_train.append(state_0_train[i])
    state_pos_train.append(state_1_train[i])
X_train['state_0'] = state_neg_train
X_train['state_1'] = state_pos_train

#========================================add to test df
state_neg_test = []
state_pos_test = []
for i in X_test['school_state']:
    state_neg_test.append(state_0_test[i])
    state_pos_test.append(state_1_test[i])
X_test['state_0'] = state_neg_test
X_test['state_1'] = state_pos_test

#========================================add to cv df
state_neg_cv = []
state_pos_cv = []
for i in X_cv['school_state']:
    state_neg_cv.append(state_0_cv[i])
    state_pos_cv.append(state_1_cv[i])
X_cv['state_0'] = state_neg_cv
X_cv['state_1'] = state_pos_cv
```

```
{'AK': 0.16470588235294117, 'AL': 0.14074074074074075, 'AR': 0.14782608695652175, 'AZ':
0.14717741935483872, 'CA': 0.14239766081871344, 'CO': 0.15725806451612903, 'CT':
0.13725490196078433, 'DC': 0.224, 'DE': 0.06756756756756757, 'FL': 0.18227665706051874, 'GA':
0.13356164383561644, 'HI': 0.12, 'IA': 0.1888111888111888, 'ID': 0.16666666666666666, 'IL':
0.16387337057728119, 'IN': 0.157439446366782, 'KS': 0.13333333333333333, 'KY':
0.14893617021276595, 'LA': 0.19887429643527205, 'MA': 0.13307240704500978, 'MD':
0.168141592920354, 'ME': 0.16666666666666666, 'MI': 0.1343065693430657, 'MN': 0.14028776978417265,
'MO': 0.14634146341463414, 'MS': 0.1254125412541254, 'MT': 0.22033898305084745, 'NC':
0.14623467600700527, 'ND': 0.1111111111111111, 'NE': 0.10294117647058823, 'NH':
0.10144927536231885, 'NJ': 0.17901234567901234, 'NM': 0.1111111111111111, 'NV':
0.1399317406143447, 'NY': 0.14111178985949907, 'OH': 0.12788632326820604, 'OK':
0.1417910447761194, 'OR': 0.15891472868217055, 'PA': 0.15091463414634146, 'RI': 0.140625, 'SC':
0.13590033975084936, 'SD': 0.19047619047619047, 'TN': 0.1188118811881188, 'TX':
0.18764988009592326, 'UT': 0.15915119363395225, 'VA': 0.1319910514541387, 'VT':
0.3333333333333333, 'WA': 0.14285714285714285, 'WI': 0.13636363636363635, 'WV':
0.12605042016806722, 'WY': 0.18181818181818182}
{'AK': 0.8352941176470589, 'AL': 0.8592592592592593, 'AR': 0.8521739130434782, 'AZ':
0.8528225806451613, 'CA': 0.8576023391812866, 'CO': 0.842741935483871, 'CT': 0.8627450980392157,
'DC': 0.776, 'DE': 0.9324324324324325, 'FL': 0.8177233429394812, 'GA': 0.8664383561643836, 'HI': 0
.88, 'IA': 0.8111888111888111, 'ID': 0.8333333333333334, 'IL': 0.8361266294227188, 'IN':
0.842560553633218, 'KS': 0.8666666666666667, 'KY': 0.851063829787234, 'LA': 0.801125703564728,
'MA': 0.8669275929549902, 'MD': 0.831858407079646, 'ME': 0.8333333333333334, 'MI':
0.8656934306569343, 'MN': 0.8597122302158273, 'MO': 0.8536585365853658, 'MS': 0.8745874587458746,
'MT': 0.7796610169491526, 'NC': 0.8537653239929948, 'ND': 0.8888888888888888, 'NE':
0.8970588235294118, 'NH': 0.8985507246376812, 'NJ': 0.8209876543209876, 'NM': 0.8888888888888888,
'NV': 0.8600682593856656, 'NY': 0.8588882101405009, 'OH': 0.872113676731794, 'OK':
0.8582089552238806, 'OR': 0.8410852713178295, 'PA': 0.8490853658536586, 'RI': 0.859375, 'SC':
0.8640996602491506, 'SD': 0.8095238095238095, 'TN': 0.8811881188118812, 'TX': 0.8123501199040767,
'UT': 0.8408488063660478, 'VA': 0.8680089485458613, 'VT': 0.6666666666666666, 'WA':
0.8571428571428571, 'WI': 0.8636363636363636, 'WV': 0.8739495798319328, 'WY': 0.8181818181818182}
{'AK': 0.20408163265306123, 'AL': 0.168141592920354, 'AR': 0.16891891891891891, 'AZ': 0.17421602787
456447, 'CA': 0.14085820895522388, 'CO': 0.14965986394557823, 'CT': 0.13991769547325103, 'DC':
0.21052631578947367, 'DE': 0.13333333333333333, 'FL': 0.13686806411837238, 'GA':
0.15129151291512916, 'HI': 0.18055555555555555, 'IA': 0.14285714285714285, 'ID':
0.1111111111111111, 'IL': 0.15472312703583063, 'IN': 0.15426997245179064, 'KS':
0.14736842105263157, 'KY': 0.16025641025641027, 'LA': 0.16382252559726962, 'MA':
0.1282798833819242, 'MD': 0.1267605633802817, 'ME': 0.10606060606060606, 'MI':
0.16556291390728478, 'MN': 0.14556962025316456, 'MO': 0.1404494382022472, 'MS':
0.18518518518518517, 'MT': 0.0967741935483871, 'NC': 0.15339233038348082, 'ND':
0.06666666666666667, 'NE': 0.1794871794871795, 'NH': 0.20930232558139536, 'NJ':
0.15355805243445692, 'NM': 0.13186813186813187, 'NV': 0.14358974358974358, 'NY':
0.13346228239845262, 'OH': 0.11455108359133127, 'OK': 0.15606936416184972, 'OR':
0.13903743315508021, 'PA': 0.15348837209302327, 'RI': 0.175, 'SC': 0.14814814814814814, 'SD':
0.10526315789473684, 'TN': 0.19457013574660634, 'TX': 0.19103313840155944, 'UT':
```

```
0.1271186440677966, 'VA': 0.13745704467353953, 'VT': 0.18181818181818182, 'WA':
0.1365079365079365, 'WI': 0.1746031746031746, 'WV': 0.2, 'WY': 0.17647058823529413}
{'AK': 0.7959183673469388, 'AL': 0.831858407079646, 'AR': 0.831081081081081, 'AZ':
0.8257839721254355, 'CA': 0.8591417910447762, 'CO': 0.8503401360544217, 'CT': 0.8600823045267489,
'DC': 0.7894736842105263, 'DE': 0.8666666666666667, 'FL': 0.8631319358816276, 'GA':
0.8487084870848709, 'HI': 0.8194444444444444, 'IA': 0.8571428571428571, 'ID': 0.8888888888888888,
'IL': 0.8452768729641694, 'IN': 0.8457300275482094, 'KS': 0.8526315789473684, 'KY':
0.8397435897435898, 'LA': 0.8361774744027304, 'MA': 0.8717201166180758, 'MD': 0.8732394366197183,
'ME': 0.8939393939393939, 'MI': 0.8344370860927153, 'MN': 0.8544303797468354, 'MO':
0.8595505617977528, 'MS': 0.8148148148148148, 'MT': 0.9032258064516129, 'NC': 0.8466076696165191,
'ND': 0.9333333333333333, 'NE': 0.8205128205128205, 'NH': 0.7906976744186046, 'NJ':
0.846441947565543, 'NM': 0.8681318681318682, 'NV': 0.8564102564102564, 'NY': 0.8665377176015474, '
OH': 0.8854489164086687, 'OK': 0.8439306358381503, 'OR': 0.8609625668449198, 'PA':
0.8465116279069768, 'RI': 0.825, 'SC': 0.8518518518518519, 'SD': 0.8947368421052632, 'TN':
0.8054298642533937, 'TX': 0.8089668615984406, 'UT': 0.8728813559322034, 'VA': 0.8625429553264605,
'VT': 0.8181818181818182, 'WA': 0.8634920634920635, 'WI': 0.8253968253968254, 'WV': 0.8, 'WY':
0.8235294117647058}
{'AK': 0.14705882352941177, 'AL': 0.15527950310559005, 'AR': 0.16666666666666666, 'AZ':
0.15609756097560976, 'CA': 0.13729246487867178, 'CO': 0.168141592920354, 'CT': 0.14375, 'DC':
0.18867924528301888, 'DE': 0.1891891891891892, 'FL': 0.14590163934426228, 'GA':
0.1421188630490956, 'HI': 0.09302325581395349, 'IA': 0.10204081632653061, 'ID':
0.1864406779661017, 'IL': 0.12408759124087591, 'IN': 0.16666666666666666, 'KS':
0.22950819672131148, 'KY': 0.11811023622047244, 'LA': 0.19246861924686193, 'MA':
0.15021459227467812, 'MD': 0.17054263565891473, 'ME': 0.24390243902439024, 'MI':
0.1619718309859155, 'MN': 0.12264150943396226, 'MO': 0.144404332129963 9, 'MS':
0.24806201550387597, 'MT': 0.12, 'NC': 0.12340425531914893, 'ND': 0.07142857142857142, 'NE':
0.1724137931034483, 'NH': 0.09090909090909091, 'NJ': 0.12698412698412698, 'NM':
0.14893617021276595, 'NV': 0.1791044776119403, 'NY': 0.14285714285714285, 'OH':
0.1542056074766355, 'OK': 0.16279069767441862, 'OR': 0.1619047619047619, 'PA':
0.12389380530973451, 'RI': 0.16666666666666666, 'SC': 0.12435233160621761, 'SD': 0.21875, 'TN':
0.18354430379746836, 'TX': 0.19336219336219337, 'UT': 0.20555555555555555, 'VA':
0.12631578947368421, 'VT': 0.2, 'WA': 0.1308016877637131, 'WI': 0.15025906735751296, 'WV':
0.16279069767441862, 'WY': 0.2}
{'AK': 0.8529411764705882, 'AL': 0.84472049689441, 'AR': 0.8333333333333334, 'AZ':
0.8439024390243902, 'CA': 0.8627075351213283, 'CO': 0.831858407079646, 'CT': 0.85625, 'DC':
0.8113207547169812, 'DE': 0.8108108108108109, 'FL': 0.8540983606557377, 'GA': 0.8578811369509044,
'HI': 0.9069767441860465, 'IA': 0.8979591836734694, 'ID': 0.8135593220338984, 'IL':
0.8759124087591241, 'IN': 0.8333333333333334, 'KS': 0.7704918032786885, 'KY': 0.8818897637795275,
'LA': 0.8075313807531381, 'MA': 0.8497854077253219, 'MD': 0.8294573643410853, 'ME':
0.7560975609756098, 'MI': 0.8380281690140845, 'MN': 0.8773584905660378, 'MO': 0.855595667870036,
'MS': 0.751937984496124, 'MT': 0.88, 'NC': 0.8765957446808511, 'ND': 0.9285714285714286, 'NE':
0.8275862068965517, 'NH': 0.9090909090909091, 'NJ': 0.873015873015873, 'NM': 0.851063829787234,
'NV': 0.8208955223880597, 'NY': 0.8571428571428571, 'OH': 0.8457943925233645, 'OK':
0.8372093023255814, 'OR': 0.8380952380952381, 'PA': 0.8761061946902655, 'RI': 0.8333333333333334,
'SC': 0.8756476683937824, 'SD': 0.78125, 'TN': 0.8164556962025317, 'TX': 0.8066378066378066, 'UT':
0.7944444444444444, 'VA': 0.8736842105263158, 'VT': 0.8, 'WA': 0.869198312236287, 'WI':
0.8497409326424871, 'WV': 0.8372093023255814, 'WY': 0.8}
```

Vectorize the Categorical Features - teacher prefix

In [34]:

```
#Teacher Prefix
prefix_0_train = get_response(X_train['teacher_prefix'],y_train)[0]
prefix_1_train = get_response(X_train['teacher_prefix'],y_train)[1]

prefix_0_test = get_response(X_test['teacher_prefix'],y_test)[0]
prefix_1_test = get_response(X_test['teacher_prefix'],y_test)[1]

prefix_0_cv = get_response(X_cv['teacher_prefix'],y_cv)[0]
prefix_1_cv = get_response(X_cv['teacher_prefix'],y_cv)[1]

print(prefix_0_train)
print(prefix_1_train)

print(prefix_0_test)
print(prefix_1_test)

print(prefix_0_cv)
print(prefix_1_cv)
#==============================add to train df
prefix_neg_train = []
prefix_pos_train = []
for i in X_train['teacher_prefix']:
    prefix_neg_train.append(prefix_0_train[i])
```

```
    prefix_pos_train.append(prefix_1_train[i])
X_train['prefix_0'] = prefix_neg_train
X_train['prefix_1'] = prefix_pos_train
#========================================add to test df
prefix_neg_test = []
prefix_pos_test = []
for i in X_test['teacher_prefix']:
    prefix_neg_test.append(prefix_0_test[i])
    prefix_pos_test.append(prefix_1_test[i])
X_test['prefix_0'] = prefix_neg_test
X_test['prefix_1'] = prefix_pos_test

#========================================add to cv df
prefix_neg_cv = []
prefix_pos_cv = []
for i in X_cv['teacher_prefix']:
    prefix_neg_cv.append(prefix_0_cv[i])
    prefix_pos_cv.append(prefix_1_cv[i])
X_cv['prefix_0'] = prefix_neg_cv
X_cv['prefix_1'] = prefix_pos_cv
```

```
{'Dr.': 0.0, 'Mr.': 0.16854401335002087, 'Mrs.': 0.1428682532275626, 'Ms.': 0.15485655029381265, '
Teacher': 0.1950354609929078}
{'Dr.': 1.0, 'Mr.': 0.8314559866499791, 'Mrs.': 0.8571317467724374, 'Ms.': 0.8451434497061874, 'Te
acher': 0.8049645390070922}
{' ': 0.0, 'Dr.': 0.0, 'Mr.': 0.15275908479138628, 'Mrs.': 0.14132817537072856, 'Ms.':
0.1601325234676974, 'Teacher': 0.21362229102167182}
{' ': 1.0, 'Dr.': 1.0, 'Mr.': 0.8472409152086138, 'Mrs.': 0.8586718246292714, 'Ms.':
0.8398674765323026, 'Teacher': 0.7863777089783281}
{'Dr.': 0.0, 'Mr.': 0.14994934143870314, 'Mrs.': 0.14088347573168514, 'Ms.': 0.16266807276562087,
'Teacher': 0.19724770642201836}
{'Dr.': 1.0, 'Mr.': 0.8500506585612969, 'Mrs.': 0.8591165242683149, 'Ms.': 0.8373319272343791, 'Te
acher': 0.8027522935779816}
```

Vectorize the Categorical Features - project_grade_category

In [35]:

```
#Project Grade Category
grad_cat_0_train = get_response(X_train['project_grade_category'],y_train)[0]
grad_cat_1_train = get_response(X_train['project_grade_category'],y_train)[1]

grad_cat_0_test = get_response(X_test['project_grade_category'],y_test)[0]
grad_cat_1_test = get_response(X_test['project_grade_category'],y_test)[1]

grad_cat_0_cv = get_response(X_cv['project_grade_category'],y_cv)[0]
grad_cat_1_cv = get_response(X_cv['project_grade_category'],y_cv)[1]

print(grad_cat_0_train)
print(grad_cat_1_train)

print(grad_cat_0_test)
print(grad_cat_1_test)

print(grad_cat_0_cv)
print(grad_cat_1_cv)
#===============================add to train df
grade_neg_train = []
grade_pos_train = []
for i in X_train['project_grade_category']:
    grade_neg_train.append(grad_cat_0_train[i])
    grade_pos_train.append(grad_cat_1_train[i])
X_train['grade_0'] = grade_neg_train
X_train['grade_1'] = grade_pos_train
#========================================add to test df
grade_neg_test = []
grade_pos_test = []
for i in X_test['project_grade_category']:
    grade_neg_test.append(grad_cat_0_test[i])
    grade_pos_test.append(grad_cat_1_test[i])
X_test['grade_0'] = grade_neg_test
X_test['grade_1'] = grade_pos_test

#========================================add to cv df
```

```
grade_neg_cv = []
grade_pos_cv = []
for i in X_cv['project_grade_category']:
    grade_neg_cv.append(grad_cat_0_cv[i])
    grade_pos_cv.append(grad_cat_1_cv[i])
X_cv['grade_0'] = grade_neg_cv
X_cv['grade_1'] = grade_pos_cv
```

```
{'3_5': 0.1458930276981853, '6_8': 0.1509633148588018, '9_12': 0.16245928338762214, 'PreK_2': 0.15
203968012956778}
{'3_5': 0.8541069723018148, '6_8': 0.8490366851411982, '9_12': 0.8375407166123778, 'PreK_2': 0.847
9603198704322}
{'3_5': 0.14448370632116214, '6_8': 0.15908141962421712, '9_12': 0.16295811518324607, 'PreK_2': 0.
14975764666555239}
{'3_5': 0.8555162936788379, '6_8': 0.8409185803757829, '9_12': 0.8370418848167539, 'PreK_2': 0.850
2423533344476}
{'3_5': 0.14715813168261113, '6_8': 0.16737935247403787, '9_12': 0.14102564102564102, 'PreK_2': 0.
14970896391152502}
{'3_5': 0.8528418683173888, '6_8': 0.8326206475259621, '9_12': 0.8589743589743589, 'PreK_2': 0.850
291036088475}
```

In [36]:

```
X_train.columns
```

Out[36]:

```
Index(['teacher_prefix', 'school_state', 'project_submitted_datetime',
       'project_grade_category',
       'teacher_number_of_previously_posted_projects', 'clean_categories',
       'clean_subcategories', 'essay', 'Cleaned_title', 'price', 'quantity',
       'essay_count', 'title_count', 'cat_0', 'cat_1', 'subcat_0', 'subcat_1',
       'state_0', 'state_1', 'prefix_0', 'prefix_1', 'grade_0', 'grade_1'],
      dtype='object')
```

In [37]:

```
X_train.head()
```

Out[37]:

	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	teacher_number_of_previously_posted_projects
29423	Teacher	OK	2017-04-12 14:58:27	9_12	0
20294	Mrs.	UT	2017-03-01 19:14:46	3_5	1
16044	Ms.	CA	2016-09-05 22:10:48	PreK_2	0
5703	Mrs.	NY	2016-07-17 07:48:39	PreK_2	33
24290	Mrs.	CT	2016-11-21 10:35:46	3_5	21

5 rows × 23 columns

In [38]:

```
X_test.columns
```

Out[38]:

```
Index(['teacher_prefix', 'school_state', 'project_submitted_datetime',
       'project_grade_category',
       'teacher_number_of_previously_posted_projects', 'clean_categories',
       'clean_subcategories', 'essay', 'Cleaned_title', 'price', 'quantity',
       'essay_count', 'title_count', 'cat_0', 'cat_1', 'subcat_0', 'subcat_1',
       'state_0', 'state_1', 'prefix_0', 'prefix_1', 'grade_0', 'grade_1'],
      dtype='object')
```

In [39]:

```
X_test.head()
```

Out[39]:

	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	teacher_number_of_previously_posted_projects
40548	Mr.	IL	2016-10-26 20:47:16	9_12	89
25630	Mr.	NY	2017-02-16 10:15:18	9_12	24
31662	Mrs.	NC	2017-01-25 21:03:12	PreK_2	5
42019	Ms.	NY	2016-09-19 13:49:28	9_12	2
27505	Mrs.	NC	2016-06-01 07:01:31	PreK_2	1

5 rows × 23 columns

Normalize Category - Cat 0

In [40]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

normalizer.fit(X_train["cat_0"].values.reshape(-1,1))   #fit has to be done only on Train data
```

```
cat_0_train_normalized = normalizer.transform(X_train["cat_0"].values.reshape(1,-1))
cat_0_test_normalized = normalizer.transform(X_test["cat_0"].values.reshape(1,-1))
cat_0_cv_normalized = normalizer.transform(X_cv["cat_0"].values.reshape(1,-1))

#reshaping after normalizing
cat_0_train_normalized = cat_0_train_normalized.reshape(-1,1)
cat_0_test_normalized = cat_0_test_normalized.reshape(-1,1)
cat_0_cv_normalized = cat_0_cv_normalized.reshape(-1,1)

print("After vectorizations")
print(cat_0_train_normalized.shape, y_train.shape)
print(cat_0_test_normalized.shape, y_test.shape)
print(cat_0_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Normalize Category - Cat 1

In [41]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

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

cat_1_train_normalized = normalizer.transform(X_train["cat_1"].values.reshape(1,-1))
cat_1_test_normalized = normalizer.transform(X_test["cat_1"].values.reshape(1,-1))
cat_1_cv_normalized = normalizer.transform(X_cv["cat_1"].values.reshape(1,-1))

#reshaping after normalizing
cat_1_train_normalized = cat_1_train_normalized.reshape(-1,1)
cat_1_test_normalized = cat_1_test_normalized.reshape(-1,1)
cat_1_cv_normalized = cat_1_cv_normalized.reshape(-1,1)

print("After vectorizations")
print(cat_1_train_normalized.shape, y_train.shape)
print(cat_1_test_normalized.shape, y_test.shape)
print(cat_1_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Normalize Sub Category - Cat 0

In [42]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

normalizer.fit(X_train["subcat_0"].values.reshape(-1,1))  #fit has to be done only on Train data
```

```
subcat_0_train_normalized = normalizer.transform(X_train["subcat_0"].values.reshape(1,-1))
subcat_0_test_normalized = normalizer.transform(X_test["subcat_0"].values.reshape(1,-1))
subcat_0_cv_normalized = normalizer.transform(X_cv["subcat_0"].values.reshape(1,-1))

#reshaping after normalizing
subcat_0_train_normalized = subcat_0_train_normalized.reshape(-1,1)
subcat_0_test_normalized = subcat_0_test_normalized.reshape(-1,1)
subcat_0_cv_normalized = subcat_0_cv_normalized.reshape(-1,1)


print("After vectorizations")
print(subcat_0_train_normalized.shape, y_train.shape)
print(subcat_0_test_normalized.shape, y_test.shape)
print(subcat_0_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Normalize Sub Category - Cat 1

In [43]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

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

subcat_1_train_normalized = normalizer.transform(X_train["subcat_1"].values.reshape(1,-1))
subcat_1_test_normalized = normalizer.transform(X_test["subcat_1"].values.reshape(1,-1))
subcat_1_cv_normalized = normalizer.transform(X_cv["subcat_1"].values.reshape(1,-1))

#reshaping after normalizing
subcat_1_train_normalized = subcat_1_train_normalized.reshape(-1,1)
subcat_1_test_normalized = subcat_1_test_normalized.reshape(-1,1)
subcat_1_cv_normalized = subcat_1_cv_normalized.reshape(-1,1)

print("After vectorizations")
print(subcat_1_train_normalized.shape, y_train.shape)
print(subcat_1_test_normalized.shape, y_test.shape)
print(subcat_1_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Normalize State - Cat 0

In [44]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

```
normalizer.fit(X_train["state_0"].values.reshape(-1,1))   #fit has to be done only on Train data

state_0_train_normalized = normalizer.transform(X_train["state_0"].values.reshape(1,-1))
state_0_test_normalized = normalizer.transform(X_test["state_0"].values.reshape(1,-1))
state_0_cv_normalized = normalizer.transform(X_cv["state_0"].values.reshape(1,-1))

#reshaping after normalizing
state_0_train_normalized = state_0_train_normalized.reshape(-1,1)
state_0_test_normalized = state_0_test_normalized.reshape(-1,1)
state_0_cv_normalized = state_0_cv_normalized.reshape(-1,1)

print("After vectorizations")
print(state_0_train_normalized.shape, y_train.shape)
print(state_0_test_normalized.shape, y_test.shape)
print(state_0_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Normalize State - Cat 1

In [45]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

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

state_1_train_normalized = normalizer.transform(X_train["state_1"].values.reshape(1,-1))
state_1_test_normalized = normalizer.transform(X_test["state_1"].values.reshape(1,-1))
state_1_cv_normalized = normalizer.transform(X_cv["state_1"].values.reshape(1,-1))

#reshaping after normalizing
state_1_train_normalized = state_1_train_normalized.reshape(-1,1)
state_1_test_normalized = state_1_test_normalized.reshape(-1,1)
state_1_cv_normalized = state_1_cv_normalized.reshape(-1,1)

print("After vectorizations")
print(state_1_train_normalized.shape, y_train.shape)
print(state_1_test_normalized.shape, y_test.shape)
print(state_1_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Normalize Prefix - Cat 0

In [46]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

```
# array.reshape(1, -1)  if it contains a single sample.

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

prefix_0_train_normalized = normalizer.transform(X_train["prefix_0"].values.reshape(1,-1))
prefix_0_test_normalized = normalizer.transform(X_test["prefix_0"].values.reshape(1,-1))
prefix_0_cv_normalized = normalizer.transform(X_cv["prefix_0"].values.reshape(1,-1))

#reshaping after normalizing
prefix_0_train_normalized = prefix_0_train_normalized.reshape(-1,1)
prefix_0_test_normalized = prefix_0_test_normalized.reshape(-1,1)
prefix_0_cv_normalized = prefix_0_cv_normalized.reshape(-1,1)

print("After vectorizations")
print(prefix_0_train_normalized.shape, y_train.shape)
print(prefix_0_test_normalized.shape, y_test.shape)
print(prefix_0_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Normalize Prefix - Cat 1

In [47]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

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

prefix_1_train_normalized = normalizer.transform(X_train["prefix_1"].values.reshape(1,-1))
prefix_1_test_normalized = normalizer.transform(X_test["prefix_1"].values.reshape(1,-1))
prefix_1_cv_normalized = normalizer.transform(X_cv["prefix_1"].values.reshape(1,-1))

#reshaping after normalizing
prefix_1_train_normalized = prefix_1_train_normalized.reshape(-1,1)
prefix_1_test_normalized = prefix_1_test_normalized.reshape(-1,1)
prefix_1_cv_normalized = prefix_1_cv_normalized.reshape(-1,1)

print("After vectorizations")
print(prefix_1_train_normalized.shape, y_train.shape)
print(prefix_1_test_normalized.shape, y_test.shape)
print(prefix_1_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Normalize Grade - Cat 0

In [48]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
```

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

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

grade_0_train_normalized = normalizer.transform(X_train["grade_0"].values.reshape(1,-1))
grade_0_test_normalized = normalizer.transform(X_test["grade_0"].values.reshape(1,-1))
grade_0_cv_normalized = normalizer.transform(X_cv["grade_0"].values.reshape(1,-1))

#reshaping after normalizing
grade_0_train_normalized = grade_0_train_normalized.reshape(-1,1)
grade_0_test_normalized = grade_0_test_normalized.reshape(-1,1)
grade_0_cv_normalized = grade_0_cv_normalized.reshape(-1,1)


print("After vectorizations")
print(grade_0_train_normalized.shape, y_train.shape)
print(grade_0_test_normalized.shape, y_test.shape)
print(grade_0_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Normalize Grade - Cat 1

In [49]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

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

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

grade_1_train_normalized = normalizer.transform(X_train["grade_1"].values.reshape(1,-1))
grade_1_test_normalized = normalizer.transform(X_test["grade_1"].values.reshape(1,-1))
grade_1_cv_normalized = normalizer.transform(X_cv["grade_1"].values.reshape(1,-1))


#reshaping after normalizing
grade_1_train_normalized = grade_1_train_normalized.reshape(-1,1)
grade_1_test_normalized = grade_1_test_normalized.reshape(-1,1)
grade_1_cv_normalized = grade_1_cv_normalized.reshape(-1,1)

print("After vectorizations")
print(grade_1_train_normalized.shape, y_train.shape)
print(grade_1_test_normalized.shape, y_test.shape)
print(grade_1_cv_normalized.shape, y_cv.shape)
```

```
After vectorizations
(24500, 1) (24500,)
(15000, 1) (15000,)
(10500, 1) (10500,)
```

Vectorize the Numerical Features - price

In [50]:

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

```
# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.
73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above mean and variance.
X_train_price_standardized = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
X_test_price_standardized = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
X_cv_price_standardized = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
```

Vectorize the Numerical Features - quantity

In [51]:

```
quantity_scalar = Normalizer()
quantity_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data


# Now standardize the data with above maen and variance.
X_train_quantity_standardized = quantity_scalar.transform(X_train['quantity'].values.reshape(-1, 1)
)
X_test_quantity_standardized = quantity_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
X_cv_quantity_standardized = quantity_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
```

Vectorize the Numerical Features - essay count

In [52]:

```
count_scalar = Normalizer()
count_scalar.fit(X_train['essay_count'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data

# Now standardize the data with above maen and variance.
X_train_essay_count_standardized = count_scalar.transform(X_train['essay_count'].values.reshape(-1
, 1))
X_test_essay_count_standardized = count_scalar.transform(X_test['essay_count'].values.reshape(-1, 1
))
X_cv_essay_count_standardized = count_scalar.transform(X_cv['essay_count'].values.reshape(-1, 1))
```

Vectorize the Numerical Features - title count

In [53]:

```
count_scalar = Normalizer()
count_scalar.fit(X_train['title_count'].values.reshape(-1,1)) # finding the mean and standard
deviation of this data

# Now standardize the data with above maen and variance.
X_train_title_count_standardized = count_scalar.transform(X_train['title_count'].values.reshape(-1
, 1))
X_test_title_count_standardized = count_scalar.transform(X_test['title_count'].values.reshape(-1, 1
))
X_cv_title_count_standardized = count_scalar.transform(X_cv['title_count'].values.reshape(-1, 1))
```

Vectorize the Numerical Features - teacher_number_of_previously_posted_projects

In [54]:

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

```
X_train_prev_proj = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].v
alues.reshape(1, -1))

X_test_prev_proj =
normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(1, -1))
X_cv_prev_proj = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.
reshape(1, -1))

#reshaping again after normalization

X_train_prev_proj = X_train_prev_proj.reshape(-1,1)
X_test_prev_proj = X_test_prev_proj.reshape(-1,1)
X_cv_prev_proj = X_cv_prev_proj.reshape(-1,1)

print('After normalization')

print(X_train_prev_proj.shape)

print(X_test_prev_proj.shape)

print(X_cv_prev_proj.shape)
```

```
After normalization
(24500, 1)
(15000, 1)
(10500, 1)
```

Vectorizing Text data

Bag of words - essay

In [55]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['essay'])

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


print("Shape of matrix after one hot encodig ",X_train_essay_bow.shape)
```

```
Shape of matrix after one hot encodig  (24500, 5000)
```

Bag of words - cleaned title

In [56]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer = CountVectorizer(min_df=5)
vectorizer.fit(X_train['Cleaned_title'])

# we use the fitted CountVectorizer to convert the text to vector
X_train_cleaned_title_bow=vectorizer.transform(X_train['Cleaned_title'].values)
X_test_cleaned_title_bow=vectorizer.transform(X_test['Cleaned_title'].values)
X_cv_cleaned_title_bow=vectorizer.transform(X_cv['Cleaned_title'].values)


print("Shape of matrix after one hot encodig ",X_train_cleaned_title_bow.shape)
```

```
Shape of matrix after one hot encodig  (24500, 2101)
```

TFIDF vectorizer - essay

In [57]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['essay'])

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


print("Shape of matrix after one hot encodig ",X_train_essay_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (24500, 5000)
```

TFIDF vectorizer - cleaned tittle

In [58]:

```
# Similarly you can vectorize for title alsovectorizer = TfidfVectorizer(min_df=10)
vectorizer = TfidfVectorizer(min_df=5)

# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer.fit(X_train['Cleaned_title'])

# we use the fitted CountVectorizer to convert the text to vector
X_train_cleaned_title_tfidf=vectorizer.transform(X_train['Cleaned_title'].values)
X_test_cleaned_title_tfidf=vectorizer.transform(X_test['Cleaned_title'].values)
X_cv_cleaned_title_tfidf=vectorizer.transform(X_cv['Cleaned_title'].values)


print("Shape of matrix after one hot encodig ",X_train_cleaned_title_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (24500, 2101)
```

Using Pretrained Models: Avg W2V

In [59]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[59]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef
loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\',
encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n
word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        m
odel[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel =
loadGloveModel(\'glove.42B.300d.txt\')\n\n# ============================\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n#
============================\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\'
\'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the
coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus",
len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words tha
t are present in both glove vectors and our coupus",       len(inter_words),"
(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove =
set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\n
print("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python
: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pic
kle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [60]:

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

In [61]:

```
# average Word2Vec
# compute average word2vec for each review.
def avg_w2v_vectors(preprocessed_essays):
    avg_w2v_vectors_text = []; # the avg-w2v for each sentence/review is stored in this list
    for sentence in tqdm(preprocessed_essays): # for each review/sentence
        vector = np.zeros(300) # as word vectors are of zero length
        cnt_words =0; # num of words with a valid vector in the sentence/review
        for word in sentence.split(): # for each word in a review/sentence
            if word in glove_words:
                vector += model[word]
                cnt_words += 1
        if cnt_words != 0:
            vector /= cnt_words
        avg_w2v_vectors_text.append(vector)
    return avg_w2v_vectors_text
```

```
X_train_essay_w2v=avg_w2v_vectors(X_train['essay'])
X_test_essay_w2v=avg_w2v_vectors(X_test['essay'])
X_cv_essay_w2v=avg_w2v_vectors(X_cv['essay'])

X_train_cleaned_title_w2v=avg_w2v_vectors(X_train['Cleaned_title'])
X_test_cleaned_title_w2v=avg_w2v_vectors(X_test['Cleaned_title'])
X_cv_cleaned_title_w2v=avg_w2v_vectors(X_cv['Cleaned_title'])
```

```
100%|████████████████████████████████████████████████████████████████████████| 24500/24500
[00:08<00:00, 2936.27it/s]
100%|████████████████████████████████████████████████████████████████████████| 15000/15000
[00:05<00:00, 2939.82it/s]
100%|████████████████████████████████████████████████████████████████████████| 10500/10500
[00:03<00:00, 2793.95it/s]
100%|███████████████████████████████████████████████████████████████████████| 24500/24500
[00:00<00:00, 43655.75it/s]
100%|███████████████████████████████████████████████████████████████████████| 15000/15000
[00:00<00:00, 43850.14it/s]
100%|███████████████████████████████████████████████████████████████████████| 10500/10500
[00:00<00:00, 44686.99it/s]
```

Using Pretrained Models: TFIDF weighted W2V - Essay

In [62]:

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

```
31318
```

In [63]:

```
# average Word2Vec
# compute average word2vec for each review.
def tfidf_w2v_vectors(tfidf_words,preprocessed_essays):
    tfidf_w2v_vectors_text = []; # the avg-w2v for each sentence/review is stored in this list
    for sentence in tqdm(preprocessed_essays): # for each review/sentence
        vector = np.zeros(300) # as word vectors are of zero length
        tf_idf_weight =0; # num of words with a valid vector in the sentence/review
        for word in sentence.split(): # for each word in a review/sentence
            if (word in glove_words) and (word in tfidf_words):
                vec = model[word] # getting the vector for each word
                # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
                tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
the tfidf value for each word
                vector += (vec * tf_idf) # calculating tfidf weighted w2v
                tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
        tfidf_w2v_vectors_text.append(vector)
    return tfidf_w2v_vectors_text

X_train_essay_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_essay,X_train['essay'])
X_test_essay_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_essay,X_test['essay'])
X_cv_essay_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_essay,X_cv['essay'])
```

```
100%|██████████████████████████████████████████████████████████████████████████| 24500/24500 [01:
03<00:00, 386.90it/s]
100%|██████████████████████████████████████████████████████████████████████████| 15000/15000 [00:
38<00:00, 387.41it/s]
100%|██████████████████████████████████████████████████████████████████████████| 10500/10500 [00:
27<00:00, 388.60it/s]
```

Using Pretrained Models: TFIDF weighted W2V - Cleaned Title

In [64]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['Cleaned_title'])
# 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_Cleaned_title = set(tfidf_model.get_feature_names())
print(len(tfidf_words_Cleaned_title))
```

```
8258
```

In [65]:

```
X_train_cleaned_title_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_Cleaned_title,X_train['Cleaned_title'
])
X_test_cleaned_title_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_Cleaned_title,X_test['Cleaned_title']
)
X_cv_cleaned_title_tfidf_w2v=tfidf_w2v_vectors(tfidf_words_Cleaned_title,X_cv['Cleaned_title'])
```

```
100%|██████████████████████████████████████████████████████████████████████████| 24500/24500
[00:01<00:00, 21495.94it/s]
100%|██████████████████████████████████████████████████████████████████████████| 15000/15000
[00:00<00:00, 22385.83it/s]
100%|██████████████████████████████████████████████████████████████████████████| 10500/10500
[00:00<00:00, 21544.43it/s]
```

In [ ]:

Merging all the above features

BOW

In [66]:

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

# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train_bow = hstack((cat_0_train_normalized, cat_1_train_normalized, subcat_0_train_normalized, s
ubcat_1_train_normalized, state_0_train_normalized, state_1_train_normalized,
grade_0_train_normalized, grade_1_train_normalized, prefix_0_train_normalized,
prefix_1_train_normalized,X_train_price_standardized,X_train_quantity_standardized,X_train_prev_pro
j,
            X_train_essay_bow,X_train_cleaned_title_bow,X_train_essay_count_standardized,X_train_ti
tle_count_standardized
            )).toarray()


X_test_bow = hstack((cat_0_test_normalized, cat_1_test_normalized, subcat_0_test_normalized, subcat
_1_test_normalized, state_0_test_normalized, state_1_test_normalized, grade_0_test_normalized,
grade_1_test_normalized, prefix_0_test_normalized,
prefix_1_test_normalized,X_test_price_standardized,X_test_quantity_standardized,X_test_prev_proj,
            X_test_essay_bow,X_test_cleaned_title_bow,X_test_essay_count_standardized,X_test_title_
count_standardized
            )).toarray()

X_cv_bow = hstack((cat_0_cv_normalized, cat_1_cv_normalized, subcat_0_cv_normalized, subcat_1_cv_no
rmalized, state_0_cv_normalized, state_1_cv_normalized, grade_0_cv_normalized,
grade_1_cv_normalized, prefix_0_cv_normalized, prefix_1_cv_normalized,X_cv_price_standardized,X_cv
_quantity_standardized,X_cv_prev_proj,

X_cv_essay_bow,X_cv_cleaned_title_bow,X_cv_essay_count_standardized,X_cv_title_count_standardized
            )).toarray()
```

```
print(X_train_bow.shape)
print(X_cv_bow.shape)
print(X_test_bow.shape)
```

```
(24500, 7116)
(10500, 7116)
(15000, 7116)
```

TFIDF

In [67]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train_tfidf = hstack((cat_0_train_normalized, cat_1_train_normalized, subcat_0_train_normalized,
subcat_1_train_normalized, state_0_train_normalized, state_1_train_normalized,
grade_0_train_normalized, grade_1_train_normalized, prefix_0_train_normalized,
prefix_1_train_normalized,X_train_price_standardized,X_train_quantity_standardized,X_train_prev_pro
j,
            X_train_essay_tfidf,X_train_cleaned_title_tfidf,X_train_essay_count_standardized,X_trai
n_title_count_standardized
            )).toarray()
```

In [68]:

```
X_test_tfidf = hstack((cat_0_test_normalized, cat_1_test_normalized, subcat_0_test_normalized,
subcat_1_test_normalized, state_0_test_normalized, state_1_test_normalized,
grade_0_test_normalized, grade_1_test_normalized, prefix_0_test_normalized,
prefix_1_test_normalized,X_test_price_standardized,X_test_quantity_standardized,X_test_prev_proj,
            X_test_essay_tfidf,X_test_cleaned_title_tfidf,X_test_essay_count_standardized,X_test_ti
tle_count_standardized
            )).toarray()
```

In [69]:

```
X_cv_tfidf = hstack((cat_0_cv_normalized, cat_1_cv_normalized, subcat_0_cv_normalized,
subcat_1_cv_normalized, state_0_cv_normalized, state_1_cv_normalized, grade_0_cv_normalized,
grade_1_cv_normalized, prefix_0_cv_normalized, prefix_1_cv_normalized,X_cv_price_standardized,X_cv
_quantity_standardized,X_cv_prev_proj,

X_cv_essay_tfidf,X_cv_cleaned_title_tfidf,X_cv_essay_count_standardized,X_cv_title_count_standardiz
d
            )).toarray()

print(X_train_tfidf.shape)
print(X_cv_tfidf.shape)
print(X_test_tfidf.shape)
```

```
(24500, 7116)
(10500, 7116)
(15000, 7116)
```

Word2Vec

In [70]:

```
train_avg_w2v_essays_np = np.array(X_train_essay_w2v)
train_avg_w2v_titles_np = np.array(X_train_cleaned_title_w2v)

test_avg_w2v_essays_np  = np.array(X_test_essay_w2v)
test_avg_w2v_titles_np  = np.array(X_test_cleaned_title_w2v)

cv_avg_w2v_essays_np   = np.array(X_cv_essay_w2v)
cv_avg_w2v_titles_np   = np.array(X_cv_cleaned_title_w2v)
```

In [71]:

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

from scipy.sparse import coo_matrix, hstack
cat_0_train_normalized_matrix = coo_matrix(cat_0_train_normalized)
cat_1_train_normalized_matrix = coo_matrix(cat_1_train_normalized)
subcat_0_train_normalized_matrix = coo_matrix(subcat_0_train_normalized)
subcat_1_train_normalized_matrix = coo_matrix(subcat_1_train_normalized)
state_0_train_normalized_matrix = coo_matrix(state_0_train_normalized)
state_1_train_normalized_matrix = coo_matrix(state_1_train_normalized)
grade_0_train_normalized_matrix = coo_matrix(grade_0_train_normalized)
grade_1_train_normalized_matrix = coo_matrix(grade_1_train_normalized)
prefix_0_train_normalized_matrix = coo_matrix(prefix_0_train_normalized)
prefix_1_train_normalized_matrix = coo_matrix(prefix_1_train_normalized)
X_train_price_standardized_matrix = coo_matrix(X_train_price_standardized)
X_train_quantity_standardized_matrix = coo_matrix(X_train_quantity_standardized)
X_train_prev_proj_matrix = coo_matrix(X_train_prev_proj)
X_train_essay_count_standardized_matrix = coo_matrix(X_train_essay_count_standardized)
X_train_title_count_standardized_matrix = coo_matrix(X_train_title_count_standardized)
train_avg_w2v_essays_np_matrix = coo_matrix(train_avg_w2v_essays_np)
train_avg_w2v_titles_np_matrix = coo_matrix(train_avg_w2v_titles_np)
```

In [72]:

```
cat_0_test_normalized_matrix = coo_matrix(cat_0_test_normalized)
cat_1_test_normalized_matrix = coo_matrix(cat_1_test_normalized)
subcat_0_test_normalized_matrix = coo_matrix(subcat_0_test_normalized)
subcat_1_test_normalized_matrix = coo_matrix(subcat_1_test_normalized)
state_0_test_normalized_matrix = coo_matrix(state_0_test_normalized)
state_1_test_normalized_matrix = coo_matrix(state_1_test_normalized)
grade_0_test_normalized_matrix = coo_matrix(grade_0_test_normalized)
grade_1_test_normalized_matrix = coo_matrix(grade_1_test_normalized)
prefix_0_test_normalized_matrix = coo_matrix(prefix_0_test_normalized)
prefix_1_test_normalized_matrix = coo_matrix(prefix_1_test_normalized)
X_test_price_standardized_matrix = coo_matrix(X_test_price_standardized)
X_test_quantity_standardized_matrix = coo_matrix(X_test_quantity_standardized)
X_test_prev_proj_matrix = coo_matrix(X_test_prev_proj)
X_test_essay_count_standardized_matrix = coo_matrix(X_test_essay_count_standardized)
X_test_title_count_standardized_matrix = coo_matrix(X_test_title_count_standardized)
test_avg_w2v_essays_np_matrix = coo_matrix(test_avg_w2v_essays_np)
test_avg_w2v_titles_np_matrix = coo_matrix(test_avg_w2v_titles_np)
```

In [73]:

```
cat_0_cv_normalized_matrix = coo_matrix(cat_0_cv_normalized)
cat_1_cv_normalized_matrix = coo_matrix(cat_1_cv_normalized)
subcat_0_cv_normalized_matrix = coo_matrix(subcat_0_cv_normalized)
subcat_1_cv_normalized_matrix = coo_matrix(subcat_1_cv_normalized)
state_0_cv_normalized_matrix = coo_matrix(state_0_cv_normalized)
state_1_cv_normalized_matrix = coo_matrix(state_1_cv_normalized)
grade_0_cv_normalized_matrix = coo_matrix(grade_0_cv_normalized)
grade_1_cv_normalized_matrix = coo_matrix(grade_1_cv_normalized)
prefix_0_cv_normalized_matrix = coo_matrix(prefix_0_cv_normalized)
prefix_1_cv_normalized_matrix = coo_matrix(prefix_1_cv_normalized)
X_cv_price_standardized_matrix = coo_matrix(X_cv_price_standardized)
X_cv_quantity_standardized_matrix = coo_matrix(X_cv_quantity_standardized)
X_cv_prev_proj_matrix = coo_matrix(X_cv_prev_proj)
X_cv_essay_count_standardized_matrix = coo_matrix(X_cv_essay_count_standardized)
X_cv_title_count_standardized_matrix = coo_matrix(X_cv_title_count_standardized)
cv_avg_w2v_essays_np_matrix = coo_matrix(cv_avg_w2v_essays_np)
cv_avg_w2v_titles_np_matrix = coo_matrix(cv_avg_w2v_titles_np)
```

In [74]:

```
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train_w2v = hstack((cat_0_train_normalized_matrix, cat_1_train_normalized_matrix,
subcat_0_train_normalized_matrix, subcat_1_train_normalized_matrix,
state_0_train_normalized_matrix, state_1_train_normalized_matrix, grade_0_train_normalized_matrix,
grade_1_train_normalized_matrix, prefix_0_train_normalized_matrix,
prefix_1_train_normalized_matrix,X_train_price_standardized_matrix,X_train_quantity_standardized_ma
rix,X_train_prev_proj_matrix,
train_avg_w2v_essays_np_matrix,train_avg_w2v_titles_np_matrix,X_train_essay_count_standardized_mat
```

```
train_avg_w2v_essays_np_matrix,train_avg_w2v_titles_np_matrix,X_train_essay_count_standardized_matri
x,X_train_title_count_standardized_matrix
            )).toarray()


X_test_w2v = hstack((cat_0_test_normalized_matrix, cat_1_test_normalized_matrix,
subcat_0_test_normalized_matrix, subcat_1_test_normalized_matrix, state_0_test_normalized_matrix,
state_1_test_normalized_matrix, grade_0_test_normalized_matrix, grade_1_test_normalized_matrix, pr
efix_0_test_normalized_matrix, prefix_1_test_normalized_matrix,X_test_price_standardized_matrix,X_
test_quantity_standardized_matrix,X_test_prev_proj_matrix,

test_avg_w2v_essays_np_matrix,test_avg_w2v_titles_np_matrix,X_test_essay_count_standardized_matrix
,X_test_title_count_standardized_matrix
            )).toarray()

X_cv_w2v = hstack((cat_0_cv_normalized_matrix, cat_1_cv_normalized_matrix,
subcat_0_cv_normalized_matrix, subcat_1_cv_normalized_matrix, state_0_cv_normalized_matrix,
state_1_cv_normalized_matrix, grade_0_cv_normalized_matrix, grade_1_cv_normalized_matrix,
prefix_0_cv_normalized_matrix,
prefix_1_cv_normalized_matrix,X_cv_price_standardized_matrix,X_cv_quantity_standardized_matrix,X_c
v_prev_proj_matrix,

cv_avg_w2v_essays_np_matrix,cv_avg_w2v_titles_np_matrix,X_cv_essay_count_standardized_matrix,X_cv_
title_count_standardized_matrix
            )).toarray()

print(X_train_w2v.shape)
print(X_test_w2v.shape)
print(X_cv_w2v.shape)
```

```
(24500, 615)
(15000, 615)
(10500, 615)
```

TFIDF- WORD2VEC

In [75]:

```
train_tfidf_w2v_essays_np = np.array(X_train_essay_tfidf_w2v)
train_tfidf_w2v_titles_np = np.array(X_train_cleaned_title_tfidf_w2v)

test_tfidf_w2v_essays_np  = np.array(X_test_essay_tfidf_w2v)
test_tfidf_w2v_titles_np  = np.array(X_test_cleaned_title_tfidf_w2v)

cv_tfidf_w2v_essays_np  = np.array(X_cv_essay_tfidf_w2v)
cv_tfidf_w2v_titles_np  = np.array(X_cv_cleaned_title_tfidf_w2v)
```

In [76]:

```
cat_0_train_normalized_matrix = coo_matrix(cat_0_train_normalized)
cat_1_train_normalized_matrix = coo_matrix(cat_1_train_normalized)
subcat_0_train_normalized_matrix = coo_matrix(subcat_0_train_normalized)
subcat_1_train_normalized_matrix = coo_matrix(subcat_1_train_normalized)
state_0_train_normalized_matrix = coo_matrix(state_0_train_normalized)
state_1_train_normalized_matrix = coo_matrix(state_1_train_normalized)
grade_0_train_normalized_matrix = coo_matrix(grade_0_train_normalized)
grade_1_train_normalized_matrix = coo_matrix(grade_1_train_normalized)
prefix_0_train_normalized_matrix = coo_matrix(prefix_0_train_normalized)
prefix_1_train_normalized_matrix = coo_matrix(prefix_1_train_normalized)
X_train_price_standardized_matrix = coo_matrix(X_train_price_standardized)
X_train_quantity_standardized_matrix = coo_matrix(X_train_quantity_standardized)
X_train_prev_proj_matrix = coo_matrix(X_train_prev_proj)
X_train_essay_count_standardized_matrix = coo_matrix(X_train_essay_count_standardized)
X_train_title_count_standardized_matrix = coo_matrix(X_train_title_count_standardized)
train_tfidf_w2v_essays_np_matrix = coo_matrix(train_tfidf_w2v_essays_np)
train_tfidf_w2v_titles_np_matrix = coo_matrix(train_tfidf_w2v_titles_np)
```

In [77]:

```
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train_tfidf_w2v = hstack((cat_0_train_normalized_matrix, cat_1_train_normalized_matrix,
subcat_0_train_normalized_matrix, subcat_1_train_normalized_matrix,
state_0_train_normalized_matrix, state_1_train_normalized_matrix, grade_0_train_normalized_matrix
```

```
state_0_train_normalized_matrix, state_1_train_normalized_matrix, grade_0_train_normalized_matrix,
grade_1_train_normalized_matrix, prefix_0_train_normalized_matrix,
prefix_1_train_normalized_matrix,X_train_price_standardized_matrix,X_train_quantity_standardized_ma
rix,X_train_prev_proj_matrix,train_tfidf_w2v_essays_np_matrix,train_tfidf_w2v_titles_np_matrix,X_t
rain_essay_count_standardized_matrix,X_train_title_count_standardized_matrix
            )).toarray()
```

In [78]:

```
cat_0_test_normalized_matrix = coo_matrix(cat_0_test_normalized)
cat_1_test_normalized_matrix = coo_matrix(cat_1_test_normalized)
subcat_0_test_normalized_matrix = coo_matrix(subcat_0_test_normalized)
subcat_1_test_normalized_matrix = coo_matrix(subcat_1_test_normalized)
state_0_test_normalized_matrix = coo_matrix(state_0_test_normalized)
state_1_test_normalized_matrix = coo_matrix(state_1_test_normalized)
grade_0_test_normalized_matrix = coo_matrix(grade_0_test_normalized)
grade_1_test_normalized_matrix = coo_matrix(grade_1_test_normalized)
prefix_0_test_normalized_matrix = coo_matrix(prefix_0_test_normalized)
prefix_1_test_normalized_matrix = coo_matrix(prefix_1_test_normalized)
X_test_price_standardized_matrix = coo_matrix(X_test_price_standardized)
X_test_quantity_standardized_matrix = coo_matrix(X_test_quantity_standardized)
X_test_prev_proj_matrix = coo_matrix(X_test_prev_proj)
X_test_essay_count_standardized_matrix = coo_matrix(X_test_essay_count_standardized)
X_test_title_count_standardized_matrix = coo_matrix(X_test_title_count_standardized)
test_tfidf_w2v_essays_np_matrix = coo_matrix(test_tfidf_w2v_essays_np)
test_tfidf_w2v_titles_np_matrix = coo_matrix(test_tfidf_w2v_titles_np)
```

In [79]:

```
cat_0_cv_normalized_matrix = coo_matrix(cat_0_cv_normalized)
cat_1_cv_normalized_matrix = coo_matrix(cat_1_cv_normalized)
subcat_0_cv_normalized_matrix = coo_matrix(subcat_0_cv_normalized)
subcat_1_cv_normalized_matrix = coo_matrix(subcat_1_cv_normalized)
state_0_cv_normalized_matrix = coo_matrix(state_0_cv_normalized)
state_1_cv_normalized_matrix = coo_matrix(state_1_cv_normalized)
grade_0_cv_normalized_matrix = coo_matrix(grade_0_cv_normalized)
grade_1_cv_normalized_matrix = coo_matrix(grade_1_cv_normalized)
prefix_0_cv_normalized_matrix = coo_matrix(prefix_0_cv_normalized)
prefix_1_cv_normalized_matrix = coo_matrix(prefix_1_cv_normalized)
X_cv_price_standardized_matrix = coo_matrix(X_cv_price_standardized)
X_cv_quantity_standardized_matrix = coo_matrix(X_cv_quantity_standardized)
X_cv_prev_proj_matrix = coo_matrix(X_cv_prev_proj)
X_cv_essay_count_standardized_matrix = coo_matrix(X_cv_essay_count_standardized)
X_cv_title_count_standardized_matrix = coo_matrix(X_cv_title_count_standardized)
cv_tfidf_w2v_essays_np_matrix = coo_matrix(cv_tfidf_w2v_essays_np)
cv_tfidf_w2v_titles_np_matrix = coo_matrix(cv_tfidf_w2v_titles_np)
```

In [80]:

```
X_test_tfidf_w2v = hstack((cat_0_test_normalized_matrix, cat_1_test_normalized_matrix,
subcat_0_test_normalized_matrix, subcat_1_test_normalized_matrix, state_0_test_normalized_matrix,
state_1_test_normalized_matrix, grade_0_test_normalized_matrix, grade_1_test_normalized_matrix, pr
efix_0_test_normalized_matrix, prefix_1_test_normalized_matrix,X_test_price_standardized_matrix,X_
test_quantity_standardized_matrix,X_test_prev_proj_matrix,
test_tfidf_w2v_essays_np_matrix,test_tfidf_w2v_titles_np_matrix,X_test_essay_count_standardized_mat
ix,X_test_title_count_standardized_matrix
            )).toarray()

X_cv_tfidf_w2v = hstack((cat_0_cv_normalized_matrix, cat_1_cv_normalized_matrix,
subcat_0_cv_normalized_matrix, subcat_1_cv_normalized_matrix, state_0_cv_normalized_matrix,
state_1_cv_normalized_matrix, grade_0_cv_normalized_matrix, grade_1_cv_normalized_matrix,
prefix_0_cv_normalized_matrix,
prefix_1_cv_normalized_matrix,X_cv_price_standardized_matrix,X_cv_quantity_standardized_matrix,X_c
v_prev_proj_matrix,            cv_tfidf_w2v_essays_np_matrix,cv_tfidf_w2v_titles_np_matrix,X_cv_ess
ay_count_standardized_matrix,X_cv_title_count_standardized_matrix
            )).toarray()

print(X_train_tfidf_w2v.shape)
print(X_test_tfidf_w2v.shape)
print(X_cv_tfidf_w2v.shape)
```

```
(24500, 615)
(15000, 615)
(10500, 615)
```

Handle NaN Values

In [81]:

```
X_train_bow[np.where(np.isnan(X_train_bow))] =0
X_cv_bow[np.where(np.isnan(X_cv_bow))] = 0
X_test_bow[np.where(np.isnan(X_test_bow))] = 0
```

In [82]:

```
X_train_tfidf[np.where(np.isnan(X_train_tfidf))] =0
X_cv_tfidf[np.where(np.isnan(X_cv_tfidf))] = 0
X_test_tfidf[np.where(np.isnan(X_test_tfidf))] = 0
```

In [83]:

```
X_train_w2v[np.where(np.isnan(X_train_w2v))] =0
X_cv_w2v[np.where(np.isnan(X_cv_w2v))] = 0
X_test_w2v[np.where(np.isnan(X_test_w2v))] = 0
```

In [84]:

```
X_train_tfidf_w2v[np.where(np.isnan(X_train_tfidf_w2v))] =0
X_cv_tfidf_w2v[np.where(np.isnan(X_cv_tfidf_w2v))] = 0
X_test_tfidf_w2v[np.where(np.isnan(X_test_tfidf_w2v))] = 0
```

Apply RANDOM FOREST on BOW SET 1

Train model for various values

In [85]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score

summary=[]
roc_auc_score_cv_bow_dict=[]
roc_auc_score_train_bow_dict=[]

num_trees = [1,2,3,4,5]
depth = [10,20,30,40,50]

for n in tqdm(num_trees):
    for d in depth:
        #create instance of model
        dt=RandomForestClassifier(max_depth=d,n_estimators=n,n_jobs=-1)

        #Fit the model on the training set
        dt.fit(X_train_bow,y_train)

        # predict the response on the crossvalidation train
        pred_bow_cv = dt.predict_proba(X_cv_bow)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_bow_cv[:,1])

        #insert into dict
        roc_auc_score_cv_bow_dict.append([d,n,roc_auc_cv])

        # fitting the model on crossvalidation train
        dt.fit(X_train_bow, y_train)

        # predict the response on the train
        pred_bow_train = dt.predict_proba(X_train_bow)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_bow_train[:,1])
```

```
        roc_auc_train =roc_auc_score(y_train,pred_bow_train[:,1])

        #insert into dict
        roc_auc_score_train_bow_dict.append([d,n,roc_auc_train])

print(roc_auc_score_cv_bow_dict)
```

```
100%|██████████████████████████████████████████████████████████████████████████████████████| 5/5 [02
:20<00:00, 29.51s/it]
```

```
[[10, 1, 0.5208618670491841], [20, 1, 0.5733979209259875], [30, 1, 0.5438413895083961], [40, 1,
0.545775949762288], [50, 1, 0.5401888377343355], [10, 2, 0.5716646626206969], [20, 2,
0.5892204028575172], [30, 2, 0.580104077676836], [40, 2, 0.5740223005578717], [50, 2,
0.5252320994110705], [10, 3, 0.6009875874306823], [20, 3, 0.6034111807133251], [30, 3,
0.5678730803788798], [40, 3, 0.5517251431765866], [50, 3, 0.5604053200389272], [10, 4,
0.6029835690428137], [20, 4, 0.5799898778735117], [30, 4, 0.5466770088804516], [40, 4,
0.5580651450369382], [50, 4, 0.5329014450738264], [10, 5, 0.5984329335797083], [20, 5,
0.6062541323928521], [30, 5, 0.577363317818832], [40, 5, 0.5571933797268854], [50, 5,
0.5955377702035003]]
```

3D Scatter Plot

In [86]:

```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_bow_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_bow_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross Val')

trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'Train')

data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(BOW)',scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='min_samples_split'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)

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

Find best Hyper-Parameter value to train model

In [87]:

```
from numpy import array

def find_best_params(input_list):
    optimal={}

    temp=pd.DataFrame(input_list)

    print(temp)
    print("="*50)

    print("Max auc score==>", max(temp[2]))
    print("*"*50)

    print("temp[2]==>",temp[2])
    print("@"*50)

    print("temp[temp[2]==max(temp[2])]==>",temp[temp[2]==max(temp[2])])
    print("^"*50)

    print("temp[temp[2]==max(temp[2])].iloc[0][0]==>",temp[temp[2]==max(temp[2])].iloc[0][0])
    print("#"*50)

    optimal_depth=int(temp[temp[2]==max(temp[2])].iloc[0][0])

    optimal_sample=int(temp[temp[2]==max(temp[2])].iloc[0][1])

    optimal['depth']=optimal_depth

    optimal['sample']=optimal_sample

    return optimal
```

In [88]:

```
find_best_params(roc_auc_score_cv_bow_dict)
```

```
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
```

```
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
==================================================
Max auc score==> 0.6062541323928521
**************************************************
temp[2]==> 0     0.520862
1      0.573398
2      0.543841
3      0.545776
4      0.540189
5      0.571665
6      0.589220
7      0.580104
8      0.574022
9      0.525232
10     0.600988
11     0.603411
12     0.567873
13     0.551725
14     0.560405
15     0.602984
16     0.579990
17     0.546677
18     0.558065
19     0.532901
20     0.598433
21     0.606254
22     0.577363
23     0.557193
24     0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0   1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
```

Out[88]:

```
{'depth': 20, 'sample': 5}
```

Use best Hyper-Parameter value to train model

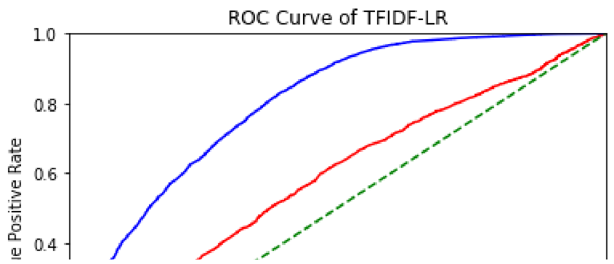
In [89]:

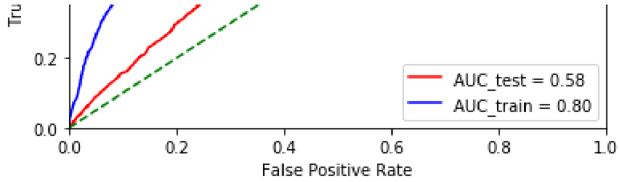
```
# train model on the best alpha
model = RandomForestClassifier(max_depth=find_best_params(roc_auc_score_cv_bow_dict)['depth'],n_es
timators=find_best_params(roc_auc_score_cv_bow_dict)['sample'])

# fitting the model on crossvalidation train
model.fit(X_train_bow, y_train)

# predict the response on the crossvalidation train
pred_bow_test = model.predict(X_test_bow)
pred_bow_train = model.predict(X_train_bow)

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_bow_test_scores=model.predict_proba(X_test_bow)
pred_bow_train_scores=model.predict_proba(X_train_bow)
```

```python
fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_bow_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_bow_train_scores[:, 1])

roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)

plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')

plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])

plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of TFIDF-LR')
plt.show()
```

```
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
==================================================
Max auc score==> 0.6062541323928521
**************************************************
temp[2]==> 0     0.520862
1     0.573398
2     0.543841
3     0.545776
4     0.540189
5     0.571665
6     0.589220
7     0.580104
8     0.574022
9     0.525232
10    0.600988
11    0.603411
12    0.567873
13    0.551725
14    0.560405
15    0.602984
16    0.579990
17    0.546677
18    0.558065
19    0.532901
20    0.598433
21    0.606254
22    0.577363
23    0.557193
24    0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
temp[temp[2]==max(temp[2])]==>      0  1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
==================================================
Max auc score==> 0.6062541323928521
**************************************************
temp[2]==> 0     0.520862
1     0.573398
2     0.543841
3     0.545776
4     0.540189
5     0.571665
6     0.589220
7     0.580104
8     0.574022
9     0.525232
10    0.600988
11    0.603411
12    0.567873
13    0.551725
14    0.560405
15    0.602984
16    0.579990
17    0.546677
18    0.558065
19    0.532901
20    0.598433
21    0.606254
22    0.577363
23    0.557193
24    0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
```



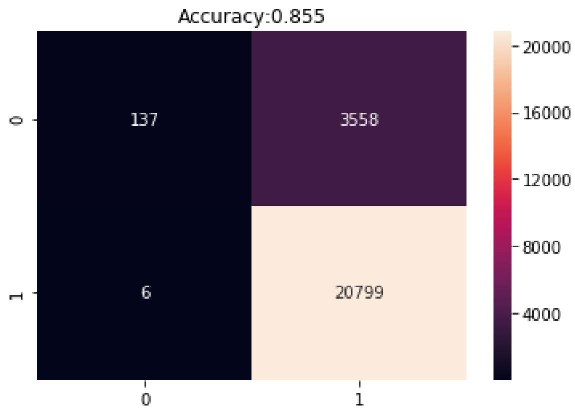


Plot Confusion Matrix

In [90]:

```
from sklearn.metrics import accuracy_score
summary = []
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix

print("Training CM for BOW")
cm =confusion_matrix(y_train, pred_bow_train, labels=None, sample_weight=None)

sns.heatmap(cm, annot=True,fmt="d")

plt.title('Accuracy:{0:.3f}'.format(accuracy_score(y_train, pred_bow_train)))

plt.show()

print("="*50)

cm =confusion_matrix(y_test, pred_bow_test, labels=None, sample_weight=None)

summary.append(['Bow_RF',find_best_params(roc_auc_score_cv_bow_dict)['depth'],find_best_params(roc
_auc_score_cv_bow_dict)['sample'],roc_auc_test])

print("="*50)
print("Testing CM for BOW")
sns.heatmap(cm, annot=True,fmt="d")
plt.title('Accuracy:{0:.3f}'.format(accuracy_score(y_test, pred_bow_test)))
plt.show()
```

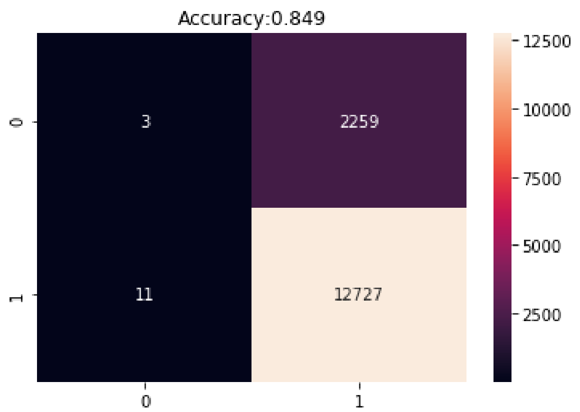
```
Training CM for BOW
```




```
==================================================
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
```

```
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
==================================================
Max auc score==> 0.6062541323928521
**************************************************
temp[2]==> 0      0.520862
1      0.573398
2      0.543841
3      0.545776
4      0.540189
5      0.571665
6      0.589220
7      0.580104
8      0.574022
9      0.525232
10     0.600988
11     0.603411
12     0.567873
13     0.551725
14     0.560405
15     0.602984
16     0.579990
17     0.546677
18     0.558065
19     0.532901
20     0.598433
21     0.606254
22     0.577363
23     0.557193
24     0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
==================================================
Max auc score==> 0.6062541323928521
**************************************************
temp[2]==> 0      0.520862
1      0.573398
```

```
1      0.573598
2      0.543841
3      0.545776
4      0.540189
5      0.571665
6      0.589220
7      0.580104
8      0.574022
9      0.525232
10     0.600988
11     0.603411
12     0.567873
13     0.551725
14     0.560405
15     0.602984
16     0.579990
17     0.546677
18     0.558065
19     0.532901
20     0.598433
21     0.606254
22     0.577363
23     0.557193
24     0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>       0   1         2
21  20   5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
==================================================
Testing CM for BOW
```



Apply RANDOM FOREST on TFIDF SET 2

Train mode for various values

In [91]:

```
roc_auc_score_cv_tfidf_dict=[]
roc_auc_score_train_tfidf_dict=[]


num_trees = [1,2,3,4,5]
depth = [10,20,30,40,50]

for n in tqdm(num_trees):
    for d in depth:
        #create instance of model
        dt=RandomForestClassifier(max_depth=d,n_estimators=n,n_jobs=-1,random_state=0)

        #Fit the model on the training set
        dt.fit(X_train_tfidf,y_train)


        # predict the response on the crossvalidation train
```

```
        # predict the response on the crossvalidation train
        pred_tfidf_cv = dt.predict_proba(X_cv_tfidf)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_cv[:,1])

        #insert into dict
        roc_auc_score_cv_tfidf_dict.append([d,n,roc_auc_cv])

         # fitting the model on crossvalidation train
        dt.fit(X_train_tfidf, y_train)

        # predict the response on the train
        pred_tfidf_train = dt.predict_proba(X_train_tfidf)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_tfidf_train[:,1])

        #insert into dict
        roc_auc_score_train_tfidf_dict.append([d,n,roc_auc_train])

print(roc_auc_score_cv_tfidf_dict)
```

```
100%|████████████████████████████████████████████████████████████████████████████████████| 5/5 [02
:09<00:00, 27.13s/it]
```

```
[[10, 1, 0.5516124665095969], [20, 1, 0.5647040712584104], [30, 1, 0.5449998232453416], [40, 1,
0.5325075549333288], [50, 1, 0.5207352696245313], [10, 2, 0.5652353270432289], [20, 2,
0.5728635126031738], [30, 2, 0.5596654654198108], [40, 2, 0.5393556821592774], [50, 2,
0.5502053719105747], [10, 3, 0.5745380770269173], [20, 3, 0.5776786778836566], [30, 3,
0.5600238275197581], [40, 3, 0.5537488246169436], [50, 3, 0.5542844018583397], [10, 4,
0.5962625351463708], [20, 4, 0.5949956045119125], [30, 4, 0.5758556962217222], [40, 4,
0.5570942695998069], [50, 4, 0.5814870854687055], [10, 5, 0.6191514841263336], [20, 5,
0.6105495893872395], [30, 5, 0.5846775601849612], [40, 5, 0.5775626361480208], [50, 5,
0.5910465370574465]]
```

3D Scatter Plot

In [92]:

```
x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_tfidf_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_tfidf_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross val')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'train')
data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(TFIDF)',scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

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

Find Best Hyper Parameter

In [93]:

```
find_best_params(roc_auc_score_cv_tfidf_dict)
```

```
     0  1         2
0   10  1  0.551612
1   20  1  0.564704
2   30  1  0.545000
3   40  1  0.532508
4   50  1  0.520735
5   10  2  0.565235
6   20  2  0.572864
7   30  2  0.559665
8   40  2  0.539356
9   50  2  0.550205
10  10  3  0.574538
11  20  3  0.577679
12  30  3  0.560024
13  40  3  0.553749
14  50  3  0.554284
15  10  4  0.596263
16  20  4  0.594996
17  30  4  0.575856
18  40  4  0.557094
19  50  4  0.581487
20  10  5  0.619151
21  20  5  0.610550
22  30  5  0.584678
23  40  5  0.577563
24  50  5  0.591047
==================================================
Max auc score==> 0.6191514841263336
**************************************************
temp[2]==> 0     0.551612
1      0.564704
2      0.545000
3      0.532508
4      0.520735
5      0.565235
6      0.572864
7      0.559665
8      0.539356
9      0.550205
```

```
10    0.574538
11    0.577679
12    0.560024
13    0.553749
14    0.554284
15    0.596263
16    0.594996
17    0.575856
18    0.557094
19    0.581487
20    0.619151
21    0.610550
22    0.584678
23    0.577563
24    0.591047
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
20  10  5  0.619151
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
```

Out[93]:

```
{'depth': 10, 'sample': 5}
```

Use Best Hyper Parameter

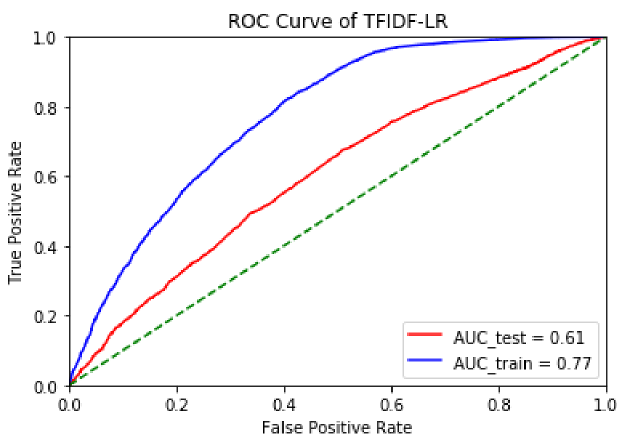
In [94]:

```
# train model on the best alpha
lr = RandomForestClassifier(max_depth=find_best_params(roc_auc_score_cv_bow_dict)
['depth'],n_estimators=find_best_params(roc_auc_score_cv_bow_dict)['sample'])

# fitting the model on crossvalidation train
lr.fit(X_train_tfidf, y_train)

# predict the response on the crossvalidation train
pred_tfidf_test = lr.predict(X_test_tfidf)
pred_tfidf_train = lr.predict(X_train_tfidf)

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_tfidf_test_scores=lr.predict_proba(X_test_tfidf)
pred_tfidf_train_scores=lr.predict_proba(X_train_tfidf)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_train_scores[:, 1])
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of TFIDF-LR')
plt.show()
```

```
    0  1         2
0  10  1  0.520862
1  20  1  0.573398
2  30  1  0.543841
3  40  1  0.545776
4  50  1  0.540189
5  10  2  0.571665
6  20  2  0.589220
7  30  2  0.580104
8  40  2  0.574022
9  50  2  0.525232
```

```
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
==================================================
Max auc score==> 0.6062541323928521
**************************************************
temp[2]==> 0     0.520862
1     0.573398
2     0.543841
3     0.545776
4     0.540189
5     0.571665
6     0.589220
7     0.580104
8     0.574022
9     0.525232
10    0.600988
11    0.603411
12    0.567873
13    0.551725
14    0.560405
15    0.602984
16    0.579990
17    0.546677
18    0.558065
19    0.532901
20    0.598433
21    0.606254
22    0.577363
23    0.557193
24    0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
==================================================
```

```
Max auc score==> 0.6062541323928521
**************************************************
temp[2]==> 0     0.520862
1     0.573398
2     0.543841
3     0.545776
4     0.540189
5     0.571665
6     0.589220
7     0.580104
8     0.574022
9     0.525232
10    0.600988
11    0.603411
12    0.567873
13    0.551725
14    0.560405
15    0.602984
16    0.579990
17    0.546677
18    0.558065
19    0.532901
20    0.598433
21    0.606254
22    0.577363
23    0.557193
24    0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
```



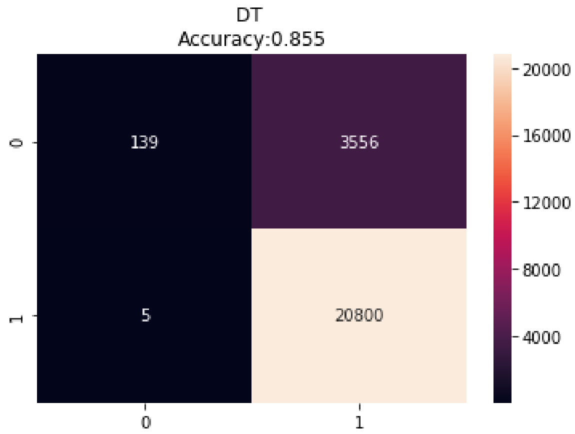
Plot Confusion Matrix

In [95]:

```
from sklearn.metrics import accuracy_score

#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for TFIDF")
cm =confusion_matrix(y_train, pred_tfidf_train, labels=None, sample_weight=None)
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT \nAccuracy:{0:.3f}'.format(accuracy_score(y_train, pred_tfidf_train)))
plt.show()

print("="*50)
print("Testing CM for TFIDF")
cm =confusion_matrix(y_test, pred_tfidf_test, labels=None, sample_weight=None)
summary.append(['Tfidf_RF',find_best_params(roc_auc_score_cv_tfidf_dict)['depth'],find_best_params
(roc_auc_score_cv_tfidf_dict)['sample'],roc_auc_test])
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT  \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, pred_tfidf_test)))
plt.show()
```

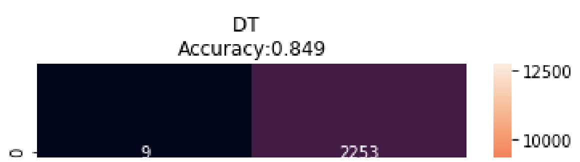
```
plt.show()
```

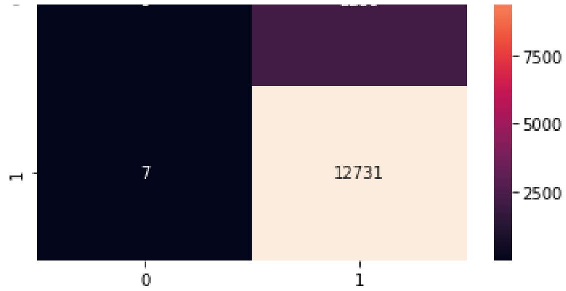
```
Training CM for TFIDF
```




```
==================================================
Testing CM for TFIDF
     0  1         2
0   10  1  0.551612
1   20  1  0.564704
2   30  1  0.545000
3   40  1  0.532508
4   50  1  0.520735
5   10  2  0.565235
6   20  2  0.572864
7   30  2  0.559665
8   40  2  0.539356
9   50  2  0.550205
10  10  3  0.574538
11  20  3  0.577679
12  30  3  0.560024
13  40  3  0.553749
14  50  3  0.554284
15  10  4  0.596263
16  20  4  0.594996
17  30  4  0.575856
18  40  4  0.557094
19  50  4  0.581487
20  10  5  0.619151
21  20  5  0.610550
22  30  5  0.584678
23  40  5  0.577563
24  50  5  0.591047
==================================================
Max auc score==> 0.6191514841263336
**************************************************
temp[2]==> 0     0.551612
1     0.564704
2     0.545000
3     0.532508
4     0.520735
5     0.565235
6     0.572864
7     0.559665
8     0.539356
9     0.550205
10    0.574538
11    0.577679
12    0.560024
13    0.553749
14    0.554284
15    0.596263
16    0.594996
17    0.575856
18    0.557094
19    0.581487
20    0.619151
21    0.610550
22    0.584678
```

```
23     0.577563
24     0.591047
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
20  10  5  0.619151
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
     0  1         2
0   10  1  0.551612
1   20  1  0.564704
2   30  1  0.545000
3   40  1  0.532508
4   50  1  0.520735
5   10  2  0.565235
6   20  2  0.572864
7   30  2  0.559665
8   40  2  0.539356
9   50  2  0.550205
10  10  3  0.574538
11  20  3  0.577679
12  30  3  0.560024
13  40  3  0.553749
14  50  3  0.554284
15  10  4  0.596263
16  20  4  0.594996
17  30  4  0.575856
18  40  4  0.557094
19  50  4  0.581487
20  10  5  0.619151
21  20  5  0.610550
22  30  5  0.584678
23  40  5  0.577563
24  50  5  0.591047
==================================================
Max auc score==> 0.6191514841263336
**************************************************
temp[2]==> 0     0.551612
1     0.564704
2     0.545000
3     0.532508
4     0.520735
5     0.565235
6     0.572864
7     0.559665
8     0.539356
9     0.550205
10    0.574538
11    0.577679
12    0.560024
13    0.553749
14    0.554284
15    0.596263
16    0.594996
17    0.575856
18    0.557094
19    0.581487
20    0.619151
21    0.610550
22    0.584678
23    0.577563
24    0.591047
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
20  10  5  0.619151
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
```





Apply RANDOM FOREST on W2V SET 3

Train mode for various values

In [96]:

```
roc_auc_score_cv_w2v_dict=[]
roc_auc_score_train_w2v_dict=[]

num_trees = [1,2,3,4,5]
depth = [10,20,30,40,50]

for n in tqdm(num_trees):
    for d in depth:
        #create instance of model
        dt=RandomForestClassifier(n_jobs=-1,max_depth=d,n_estimators=n)

        #Fit the model on the training set
        dt.fit(X_train_w2v,y_train)


        # predict the response on the crossvalidation train
        pred_w2v_cv = dt.predict_proba(X_cv_w2v)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_w2v_cv[:,1])

        #insert into dict
        roc_auc_score_cv_w2v_dict.append([d,n,roc_auc_cv])

         # fitting the model on crossvalidation train
        dt.fit(X_train_w2v, y_train)

        # predict the response on the train
        pred_w2v_train = dt.predict_proba(X_train_w2v)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_w2v_train[:,1])

        #insert into dict
        roc_auc_score_train_w2v_dict.append([d,n,roc_auc_train])

print(roc_auc_score_cv_w2v_dict)
```

```
100%|██████████████████████████████████████████████████████████████████████████████████| 5/5 [01
:02<00:00, 13.21s/it]
```

```
[[10, 1, 0.5613926311797626], [20, 1, 0.49422331063104524], [30, 1, 0.4973851291311442], [40, 1, 0
.5110134091963855], [50, 1, 0.5305005571491025], [10, 2, 0.570078936009217], [20, 2,
0.516912587064067], [30, 2, 0.515879794363843], [40, 2, 0.5033755889135794], [50, 2,
0.5244267499295638], [10, 3, 0.5850235600853552], [20, 3, 0.5301090048457697], [30, 3,
0.540439836433577], [40, 3, 0.538808026092529 9], [50, 3, 0.526065963421633], [10, 4,
0.580489218638853], [20, 4, 0.5525784891635226], [30, 4, 0.5408706006420834], [40, 4,
0.55050032903287], [50, 4, 0.5701027748639432], [10, 5, 0.597408748370864], [20, 5,
0.5593711104676943], [30, 5, 0.5490557581956601], [40, 5, 0.5374488571553863], [50, 5,
0.5662218234832344]]
```

3D Scatter Plot

3D Scatter Plot

In [97]:

```
x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_w2v_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_w2v_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross val')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'train')
data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(W2V)',scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

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

Find Best Hyper Parameter

In [98]:

```
find_best_params(roc_auc_score_cv_w2v_dict)
```

	0	1	2
0	10	1	0.561393

```
1   20  1  0.494223
2   30  1  0.497385
3   40  1  0.511013
4   50  1  0.530501
5   10  2  0.570079
6   20  2  0.516913
7   30  2  0.515880
8   40  2  0.503376
9   50  2  0.524427
10  10  3  0.585024
11  20  3  0.530109
12  30  3  0.540440
13  40  3  0.538808
14  50  3  0.526066
15  10  4  0.580489
16  20  4  0.552578
17  30  4  0.540871
18  40  4  0.550500
19  50  4  0.570103
20  10  5  0.597409
21  20  5  0.559371
22  30  5  0.549056
23  40  5  0.537449
24  50  5  0.566222
==================================================
Max auc score==> 0.597408748370864
**************************************************
temp[2]==> 0     0.561393
1     0.494223
2     0.497385
3     0.511013
4     0.530501
5     0.570079
6     0.516913
7     0.515880
8     0.503376
9     0.524427
10    0.585024
11    0.530109
12    0.540440
13    0.538808
14    0.526066
15    0.580489
16    0.552578
17    0.540871
18    0.550500
19    0.570103
20    0.597409
21    0.559371
22    0.549056
23    0.537449
24    0.566222
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0   1         2
20  10  5  0.597409
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
```

Out[98]:

```
{'depth': 10, 'sample': 5}
```

Use Best Hyper Parameter

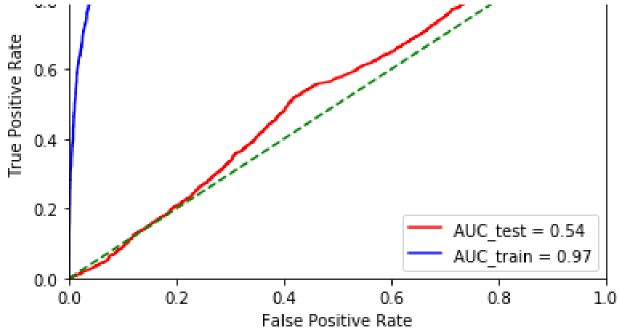
In [99]:

```
# train model on the best alpha
lr = RandomForestClassifier(max_depth=find_best_params(roc_auc_score_cv_bow_dict)
['depth'],n_estimators=find_best_params(roc_auc_score_cv_bow_dict)['sample'])

# fitting the model on crossvalidation train
lr.fit(X_train_w2v, y_train)
```

```
# predict the response on the crossvalidation train
pred_w2v_test = lr.predict(X_test_w2v)
pred_w2v_train = lr.predict(X_train_w2v)

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_w2v_test_scores=lr.predict_proba(X_test_w2v)
pred_w2v_train_scores=lr.predict_proba(X_train_w2v)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_w2v_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_w2v_train_scores[:, 1])
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of W2V-LR')
plt.show()
```

```
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
==================================================
Max auc score==> 0.6062541323928521
**************************************************
temp[2]==> 0     0.520862
1     0.573398
2     0.543841
3     0.545776
4     0.540189
5     0.571665
6     0.589220
7     0.580104
8     0.574022
9     0.525232
10    0.600988
11    0.603411
12    0.567873
13    0.551725
14    0.560405
15    0.602984
16    0.579990
17    0.546677
18    0.558065
19    0.532901
20    0.598433
21    0.606254
```

```
22    0.577363
23    0.557193
24    0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
==================================================
Max auc score==> 0.6062541323928521
**************************************************
temp[2]==> 0     0.520862
1     0.573398
2     0.543841
3     0.545776
4     0.540189
5     0.571665
6     0.589220
7     0.580104
8     0.574022
9     0.525232
10    0.600988
11    0.603411
12    0.567873
13    0.551725
14    0.560405
15    0.602984
16    0.579990
17    0.546677
18    0.558065
19    0.532901
20    0.598433
21    0.606254
22    0.577363
23    0.557193
24    0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
```

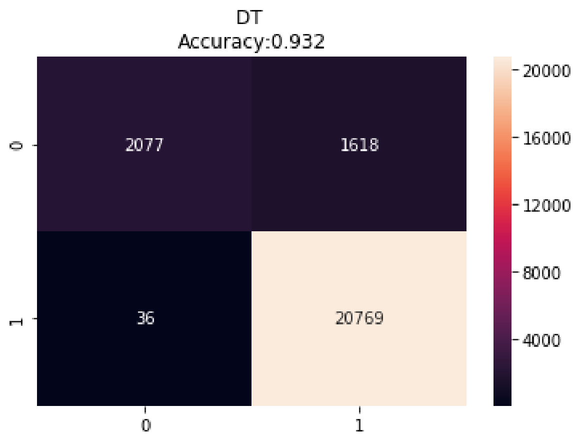


Plot Confusion Matrix

In [100]:

```
from sklearn.metrics import accuracy_score

#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for W2V")
cm =confusion_matrix(y_train, pred_w2v_train, labels=None, sample_weight=None)
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT \nAccuracy:{0:.3f}'.format(accuracy_score(y_train, pred_w2v_train)))
plt.show()
print("="*50)
print("Testing CM for W2V")
cm =confusion_matrix(y_test, pred_w2v_test, labels=None, sample_weight=None)
summary.append(['W2v_RF',find_best_params(roc_auc_score_cv_w2v_dict)['depth'],find_best_params(roc
_auc_score_cv_w2v_dict)['sample'],roc_auc_test])
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT  \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, pred_w2v_test)))
plt.show()
```

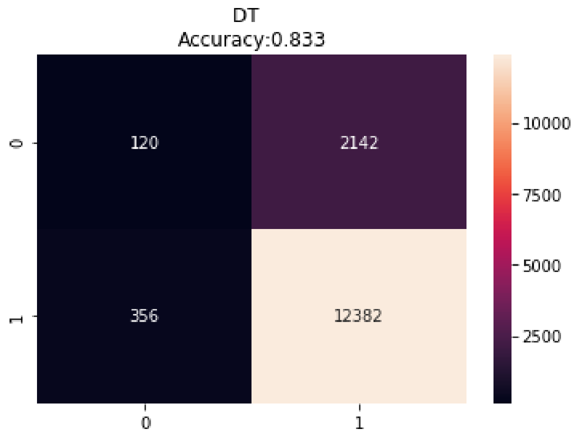
```
Training CM for W2V
```




```
==================================================
Testing CM for W2V
     0  1         2
0   10  1  0.561393
1   20  1  0.494223
2   30  1  0.497385
3   40  1  0.511013
4   50  1  0.530501
5   10  2  0.570079
6   20  2  0.516913
7   30  2  0.515880
8   40  2  0.503376
9   50  2  0.524427
10  10  3  0.585024
11  20  3  0.530109
12  30  3  0.540440
13  40  3  0.538808
14  50  3  0.526066
```

```
15  10  4  0.580489
16  20  4  0.552578
17  30  4  0.540871
18  40  4  0.550500
19  50  4  0.570103
20  10  5  0.597409
21  20  5  0.559371
22  30  5  0.549056
23  40  5  0.537449
24  50  5  0.566222
==================================================
Max auc score==> 0.597408748370864
**************************************************
temp[2]==> 0     0.561393
1     0.494223
2     0.497385
3     0.511013
4     0.530501
5     0.570079
6     0.516913
7     0.515880
8     0.503376
9     0.524427
10    0.585024
11    0.530109
12    0.540440
13    0.538808
14    0.526066
15    0.580489
16    0.552578
17    0.540871
18    0.550500
19    0.570103
20    0.597409
21    0.559371
22    0.549056
23    0.537449
24    0.566222
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
20  10  5  0.597409
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
     0  1         2
0   10  1  0.561393
1   20  1  0.494223
2   30  1  0.497385
3   40  1  0.511013
4   50  1  0.530501
5   10  2  0.570079
6   20  2  0.516913
7   30  2  0.515880
8   40  2  0.503376
9   50  2  0.524427
10  10  3  0.585024
11  20  3  0.530109
12  30  3  0.540440
13  40  3  0.538808
14  50  3  0.526066
15  10  4  0.580489
16  20  4  0.552578
17  30  4  0.540871
18  40  4  0.550500
19  50  4  0.570103
20  10  5  0.597409
21  20  5  0.559371
22  30  5  0.549056
23  40  5  0.537449
24  50  5  0.566222
==================================================
Max auc score==> 0.597408748370864
**************************************************
temp[2]==> 0     0.561393
1     0.494223
2     0.497385
```

```
3      0.511013
4      0.530501
5      0.570079
6      0.516913
7      0.515880
8      0.503376
9      0.524427
10     0.585024
11     0.530109
12     0.540440
13     0.538808
14     0.526066
15     0.580489
16     0.552578
17     0.540871
18     0.550500
19     0.570103
20     0.597409
21     0.559371
22     0.549056
23     0.537449
24     0.566222
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
20  10  5  0.597409
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
```



Apply RANDOM FOREST on TFIDFW2V SET 4

Train model for various values

In [101]:

```
roc_auc_score_cv_tfidf_w2v_dict=[]
roc_auc_score_train_tfidf_w2v_dict=[]

num_trees = [1,2,3,4,5]
depth = [10,20,30,40,50]

for n in tqdm(num_trees):
    for d in depth:
        #create instance of model
        dt=RandomForestClassifier(n_jobs=-1,max_depth=d,n_estimators=n)

        #Fit the model on the training set
        dt.fit(X_train_tfidf_w2v,y_train)


        # predict the response on the crossvalidation train
        pred_tfidf_w2v_cv = dt.predict_proba(X_cv_tfidf_w2v)

        #evaluate CV roc_auc
```

```
        roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_w2v_cv[:,1])

        #insert into dict
        roc_auc_score_cv_tfidf_w2v_dict.append([d,n,roc_auc_cv])

         # fitting the model on crossvalidation train
        dt.fit(X_train_tfidf_w2v, y_train)

        # predict the response on the train
        pred_tfidf_w2v_train = dt.predict_proba(X_train_tfidf_w2v)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_tfidf_w2v_train[:,1])

        #insert into dict
        roc_auc_score_train_tfidf_w2v_dict.append([d,n,roc_auc_train])

print(roc_auc_score_cv_tfidf_w2v_dict)
```

```
100%|██████████████████████████████████████████████████████████████████████████████████| 5/5 [00
:58<00:00, 12.26s/it]
```

```
[[10, 1, 0.5648046691000481], [20, 1, 0.5151248854902561], [30, 1, 0.5092125307221912], [40, 1,
0.5347025715004472], [50, 1, 0.5225441179981511], [10, 2, 0.5918376824070881], [20, 2,
0.5272408682840579], [30, 2, 0.5151356891316997], [40, 2, 0.5230183447248582], [50, 2,
0.5403331814683757], [10, 3, 0.5699472024271567], [20, 3, 0.5335467943966435], [30, 3,
0.5350110243191032], [40, 3, 0.5426292209384347], [50, 3, 0.5503429500855471], [10, 4,
0.5828431018678539], [20, 4, 0.5434811146325865], [30, 4, 0.5446078813024813], [40, 4,
0.5407925310494884], [50, 4, 0.5409605011076035], [10, 5, 0.6010921525111453], [20, 5,
0.5514627032439474], [30, 5, 0.5518608439974719], [40, 5, 0.539511750500917], [50, 5,
0.5823874361513647]]
```

3D Scatter Plot

In [102]:

```
x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_tfidf_w2v_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_tfidf_w2v_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross val')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'train')
data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(TFIDF_W2V)',scene = dict(
        xaxis = dict(title='max_depth'),
        yaxis = dict(title='min_samples_split'),
        zaxis = dict(title='AUC'),))

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

Find best Hyper-Parameter value to train model

In [103]:

```
find_best_params(roc_auc_score_cv_tfidf_w2v_dict)
```

```
     0  1         2
0   10  1  0.564805
1   20  1  0.515125
2   30  1  0.509213
3   40  1  0.534703
4   50  1  0.522544
5   10  2  0.591838
6   20  2  0.527241
7   30  2  0.515136
8   40  2  0.523018
9   50  2  0.540333
10  10  3  0.569947
11  20  3  0.533547
12  30  3  0.535011
13  40  3  0.542629
14  50  3  0.550343
15  10  4  0.582843
16  20  4  0.543481
17  30  4  0.544608
18  40  4  0.540793
19  50  4  0.540961
20  10  5  0.601092
21  20  5  0.551463
22  30  5  0.551861
23  40  5  0.539512
24  50  5  0.582387
==================================================
Max auc score==> 0.6010921525111453
**************************************************
temp[2]==> 0     0.564805
1      0.515125
2      0.509213
3      0.534703
4      0.522544
5      0.591838
6      0.527241
7      0.515136
8      0.523018
9      0.540333
10     0.569947
11     0.533547
12     0.535011
13     0.542629
```

```
14    0.550343
15    0.582843
16    0.543481
17    0.544608
18    0.540793
19    0.540961
20    0.601092
21    0.551463
22    0.551861
23    0.539512
24    0.582387
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
20  10  5  0.601092
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
###################################################
```

Out[103]:

```
{'depth': 10, 'sample': 5}
```

Use best Hyper-Parameter value to train model

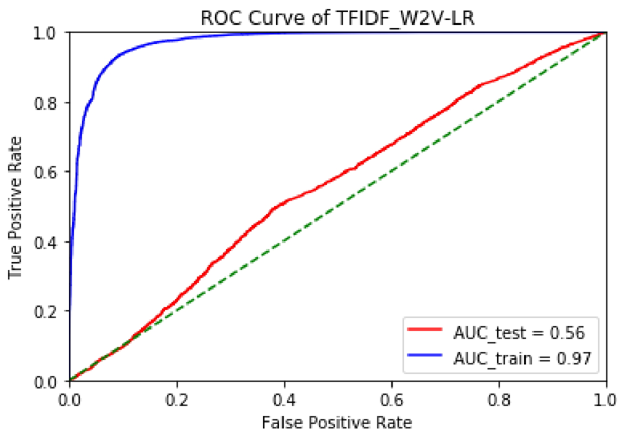
In [104]:

```python
# train model on the best alpha
lr = RandomForestClassifier(max_depth=find_best_params(roc_auc_score_cv_bow_dict)
['depth'],n_estimators=find_best_params(roc_auc_score_cv_bow_dict)['sample'])

# fitting the model on crossvalidation train
lr.fit(X_train_tfidf_w2v, y_train)

# predict the response on the crossvalidation train
pred_tfidf_w2v_test = lr.predict(X_test_tfidf_w2v)
pred_tfidf_w2v_train = lr.predict(X_train_tfidf_w2v)

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_tfidf_w2v_test_scores=lr.predict_proba(X_test_tfidf_w2v)
pred_tfidf_w2v_train_scores=lr.predict_proba(X_train_tfidf_w2v)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_w2v_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_w2v_train_scores[:, 1])

roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)

plt.title('Receiver Operating Characteristic')

plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of TFIDF_W2V-LR')
plt.show()
```

```
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
```

```
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
=================================================
Max auc score==> 0.6062541323928521
*************************************************
temp[2]==> 0     0.520862
1     0.573398
2     0.543841
3     0.545776
4     0.540189
5     0.571665
6     0.589220
7     0.580104
8     0.574022
9     0.525232
10    0.600988
11    0.603411
12    0.567873
13    0.551725
14    0.560405
15    0.602984
16    0.579990
17    0.546677
18    0.558065
19    0.532901
20    0.598433
21    0.606254
22    0.577363
23    0.557193
24    0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
#################################################
     0  1         2
0   10  1  0.520862
1   20  1  0.573398
2   30  1  0.543841
3   40  1  0.545776
4   50  1  0.540189
5   10  2  0.571665
6   20  2  0.589220
7   30  2  0.580104
8   40  2  0.574022
9   50  2  0.525232
10  10  3  0.600988
11  20  3  0.603411
12  30  3  0.567873
13  40  3  0.551725
14  50  3  0.560405
15  10  4  0.602984
16  20  4  0.579990
17  30  4  0.546677
18  40  4  0.558065
19  50  4  0.532901
20  10  5  0.598433
21  20  5  0.606254
22  30  5  0.577363
23  40  5  0.557193
24  50  5  0.595538
=================================================
Max auc score==> 0.6062541323928521
```

```
max auc score==> 0.6062541525926521
**************************************************
temp[2]==> 0      0.520862
1      0.573398
2      0.543841
3      0.545776
4      0.540189
5      0.571665
6      0.589220
7      0.580104
8      0.574022
9      0.525232
10     0.600988
11     0.603411
12     0.567873
13     0.551725
14     0.560405
15     0.602984
16     0.579990
17     0.546677
18     0.558065
19     0.532901
20     0.598433
21     0.606254
22     0.577363
23     0.557193
24     0.595538
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0   1         2
21  20  5  0.606254
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
```



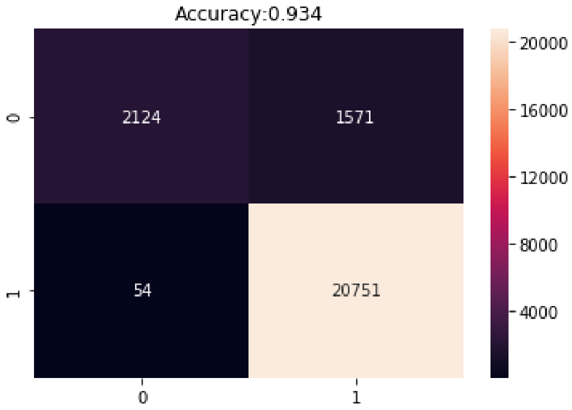
Plot Confusion Matrix

In [105]:

```python
from sklearn.metrics import accuracy_score

#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for TFIDF_W2V")
cm =confusion_matrix(y_train, pred_tfidf_w2v_train, labels=None, sample_weight=None)
sns.heatmap(cm, annot=True,fmt="d")
plt.title('Accuracy:{0:.3f}'.format(accuracy_score(y_train, pred_tfidf_w2v_train)))
plt.show()

print("="*50)
print("Testing CM for TFIDF_W2V")

cm =confusion_matrix(y_test, pred_tfidf_w2v_test, labels=None, sample_weight=None)
summary.append(['Tfidf_w2v_RF',find_best_params(roc_auc_score_cv_tfidf_w2v_dict)
['depth'],find_best_params(roc_auc_score_cv_tfidf_w2v_dict)['sample'],roc_auc_test])
sns.heatmap(cm, annot=True,fmt="d")
```

```
print("="*50)
print("Testing CM for TFIDF_W2V")
plt.title('Accuracy:{0:.3f}'.format(accuracy_score(y_test, pred_tfidf_w2v_test)))
plt.show()
```

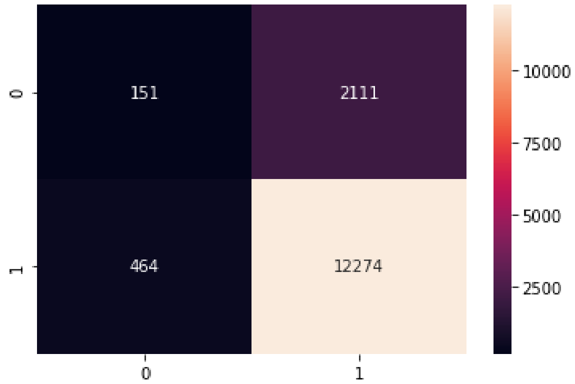
```
Training CM for TFIDF_W2V
```




```
==================================================
Testing CM for TFIDF_W2V
     0  1         2
0   10  1  0.564805
1   20  1  0.515125
2   30  1  0.509213
3   40  1  0.534703
4   50  1  0.522544
5   10  2  0.591838
6   20  2  0.527241
7   30  2  0.515136
8   40  2  0.523018
9   50  2  0.540333
10  10  3  0.569947
11  20  3  0.533547
12  30  3  0.535011
13  40  3  0.542629
14  50  3  0.550343
15  10  4  0.582843
16  20  4  0.543481
17  30  4  0.544608
18  40  4  0.540793
19  50  4  0.540961
20  10  5  0.601092
21  20  5  0.551463
22  30  5  0.551861
23  40  5  0.539512
24  50  5  0.582387
==================================================
Max auc score==> 0.6010921525111453
**************************************************
temp[2]==> 0     0.564805
1     0.515125
2     0.509213
3     0.534703
4     0.522544
5     0.591838
6     0.527241
7     0.515136
8     0.523018
9     0.540333
10    0.569947
11    0.533547
12    0.535011
13    0.542629
14    0.550343
15    0.582843
16    0.543481
17    0.544608
18    0.540793
19    0.540961
20    0.601092
```

```
21    0.551463
22    0.551861
23    0.539512
24    0.582387
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
20  10  5  0.601092
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
     0  1         2
0   10  1  0.564805
1   20  1  0.515125
2   30  1  0.509213
3   40  1  0.534703
4   50  1  0.522544
5   10  2  0.591838
6   20  2  0.527241
7   30  2  0.515136
8   40  2  0.523018
9   50  2  0.540333
10  10  3  0.569947
11  20  3  0.533547
12  30  3  0.535011
13  40  3  0.542629
14  50  3  0.550343
15  10  4  0.582843
16  20  4  0.543481
17  30  4  0.544608
18  40  4  0.540793
19  50  4  0.540961
20  10  5  0.601092
21  20  5  0.551463
22  30  5  0.551861
23  40  5  0.539512
24  50  5  0.582387
==================================================
Max auc score==> 0.6010921525111453
**************************************************
temp[2]==> 0     0.564805
1     0.515125
2     0.509213
3     0.534703
4     0.522544
5     0.591838
6     0.527241
7     0.515136
8     0.523018
9     0.540333
10    0.569947
11    0.533547
12    0.535011
13    0.542629
14    0.550343
15    0.582843
16    0.543481
17    0.544608
18    0.540793
19    0.540961
20    0.601092
21    0.551463
22    0.551861
23    0.539512
24    0.582387
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
20  10  5  0.601092
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 10.0
##################################################
==================================================
Testing CM for TFIDF_W2V
```

Accuracy:0.828



Apply GBDT on BOW SET 1

In [106]:

```
# Considering 10k points as more points results in system getting hang
X_train_bow = X_train_bow[:10000]
X_test_bow = X_test_bow[:3000]
X_cv_bow = X_cv_bow[:3000]
```

In [107]:

```
X_train_w2v = X_train_w2v[:10000]
X_test_w2v = X_test_w2v[:3000]
X_cv_w2v = X_cv_w2v[:3000]
```

In [108]:

```
X_train_tfidf = X_train_tfidf[:10000]
X_test_tfidf = X_test_tfidf[:3000]
X_cv_tfidf = X_cv_tfidf[:3000]
```

In [109]:

```
X_train_tfidf_w2v = X_train_tfidf_w2v[:10000]
X_test_tfidf_w2v = X_test_tfidf_w2v[:3000]
X_cv_tfidf_w2v = X_cv_tfidf_w2v[:3000]
```

In [110]:

```
y_train = y_train[:10000]
y_cv = y_cv[:3000]
y_test = y_test[:3000]
```

Train model for various values

In [111]:

```
from xgboost import XGBClassifier


roc_auc_score_cv_bow_dict=[]
roc_auc_score_train_bow_dict=[]


num_trees = [1,2,3,4,5,6,7,8,9,10]
max_depth=[1, 5, 10, 20]


for n in tqdm(num_trees):
    for d in max_depth:
        #create instance of model
        dt=XGBClassifier(max_depth=d,n_estimators=n,learning_rate=0.01,nthread=1)
```

```
        #Fit the model on the training set
        dt.fit(X_train_bow,y_train)


        # predict the response on the crossvalidation train
        pred_bow_cv = dt.predict_proba(X_cv_bow)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_bow_cv[:,1])

        #insert into dict
        roc_auc_score_cv_bow_dict.append([d,n,roc_auc_cv])

         # fitting the model on crossvalidation train
        dt.fit(X_train_bow, y_train)

        # predict the response on the train
        pred_bow_train = dt.predict_proba(X_train_bow)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_bow_train[:,1])

        #insert into dict
        roc_auc_score_train_bow_dict.append([d,n,roc_auc_train])

print(roc_auc_score_cv_bow_dict)
```

```
100%|██████████████████████████████████████████████████████████████████| 10/10
[49:55<00:00, 397.33s/it]
```

```
[[1, 1, 0.5624898020232786], [5, 1, 0.6039441524801479], [10, 1, 0.5691422821712172], [20, 1,
0.5223280451158491], [1, 2, 0.5624898020232786], [5, 2, 0.6086199248069183], [10, 2,
0.5663463368867617], [20, 2, 0.5171270769879256], [1, 3, 0.5624898020232786], [5, 3,
0.600439022897857], [10, 3, 0.568026878467312], [20, 3, 0.5143935943108887], [1, 4,
0.5825177784727511], [5, 4, 0.6125384973621233], [10, 4, 0.5668358397693898], [20, 4,
0.515154193408278], [1, 5, 0.5825177784727511], [5, 5, 0.6140860403295986], [10, 5,
0.5662876985206134], [20, 5, 0.5148414554552376], [1, 6, 0.5825177784727511], [5, 6,
0.6119652860872402], [10, 6, 0.5675148550527576], [20, 6, 0.5220807441803546], [1, 7,
0.5825177784727511], [5, 7, 0.6128508103992167], [10, 7, 0.566653975851191], [20, 7,
0.5297181789133036], [1, 8, 0.5825177784727511], [5, 8, 0.6126566239258131], [10, 8,
0.5679707895953443], [20, 8, 0.5429708745784836], [1, 9, 0.5825177784727511], [5, 9,
0.6030807237844013], [10, 9, 0.5679597417872295], [20, 9, 0.5433269539323398], [1, 10,
0.5825177784727511], [5, 10, 0.6113342862776023], [10, 10, 0.5689115529478951], [20, 10,
0.5464301132655282]]
```

3D Scatter Plot

In [112]:

```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()

x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_bow_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_bow_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross val')
```

```
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'train')
data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(BOW)',scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='learning_rate'),
        zaxis = dict(title='AUC'),))

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

Find best Hper-parameter to train the model

In [113]:

```
find_best_params(roc_auc_score_cv_bow_dict)
```

	0	1	2
0	1	1	0.562490
1	5	1	0.603944
2	10	1	0.569142
3	20	1	0.522328
4	1	2	0.562490
5	5	2	0.608620
6	10	2	0.566346
7	20	2	0.517127
8	1	3	0.562490
9	5	3	0.600439
10	10	3	0.568027
11	20	3	0.514394
12	1	4	0.582518
13	5	4	0.612538
14	10	4	0.566836
15	20	4	0.515154
16	1	5	0.582518
17	5	5	0.614086
18	10	5	0.566288
19	20	5	0.514841
20	1	6	0.582518
21	5	6	0.611965
22	10	6	0.567515
23	20	6	0.522081

```
24   1   7  0.582518
25   5   7  0.612851
26  10   7  0.566654
27  20   7  0.529718
28   1   8  0.582518
29   5   8  0.612657
30  10   8  0.567971
31  20   8  0.542971
32   1   9  0.582518
33   5   9  0.603081
34  10   9  0.567960
35  20   9  0.543327
36   1  10  0.582518
37   5  10  0.611334
38  10  10  0.568912
39  20  10  0.546430
==================================================
Max auc score==> 0.6140860403295986
**************************************************
temp[2]==> 0     0.562490
1      0.603944
2      0.569142
3      0.522328
4      0.562490
5      0.608620
6      0.566346
7      0.517127
8      0.562490
9      0.600439
10     0.568027
11     0.514394
12     0.582518
13     0.612538
14     0.566836
15     0.515154
16     0.582518
17     0.614086
18     0.566288
19     0.514841
20     0.582518
21     0.611965
22     0.567515
23     0.522081
24     0.582518
25     0.612851
26     0.566654
27     0.529718
28     0.582518
29     0.612657
30     0.567971
31     0.542971
32     0.582518
33     0.603081
34     0.567960
35     0.543327
36     0.582518
37     0.611334
38     0.568912
39     0.546430
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
17  5  5  0.614086
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```

Out[113]:

```
{'depth': 5, 'sample': 5}
```

Use best Hper-parameter to train the model

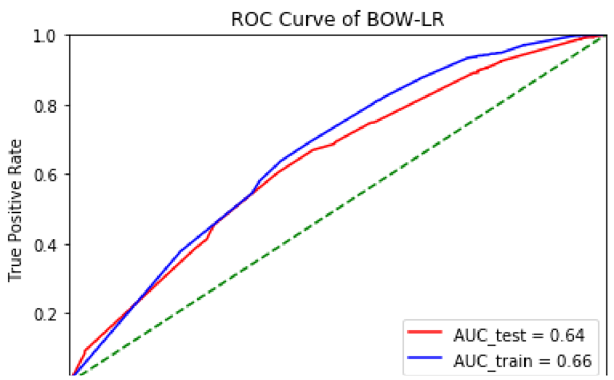
In [114]:

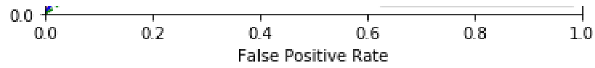
```
# train model on the best alpha
lr = XGBClassifier(learning_rate=0.01,max_depth=find_best_params(roc_auc_score_cv_bow_dict)
['depth'],n_estimators=find_best_params(roc_auc_score_cv_bow_dict)['sample'])

# fitting the model on crossvalidation train
lr.fit(X_train_bow, y_train)

# predict the response on the crossvalidation train
pred_bow_test = lr.predict(X_test_bow)
pred_bow_train = lr.predict(X_train_bow)

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_bow_test_scores=lr.predict_proba(X_test_bow)
pred_bow_train_scores=lr.predict_proba(X_train_bow)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_bow_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_bow_train_scores[:, 1])
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of BOW-LR')
plt.show()
```

```
     0   1         2
0    1   1  0.562490
1    5   1  0.603944
2   10   1  0.569142
3   20   1  0.522328
4    1   2  0.562490
5    5   2  0.608620
6   10   2  0.566346
7   20   2  0.517127
8    1   3  0.562490
9    5   3  0.600439
10  10   3  0.568027
11  20   3  0.514394
12   1   4  0.582518
13   5   4  0.612538
14  10   4  0.566836
15  20   4  0.515154
16   1   5  0.582518
17   5   5  0.614086
18  10   5  0.566288
19  20   5  0.514841
20   1   6  0.582518
21   5   6  0.611965
22  10   6  0.567515
23  20   6  0.522081
24   1   7  0.582518
25   5   7  0.612851
26  10   7  0.566654
27  20   7  0.529718
28   1   8  0.582518
29   5   8  0.612657
30  10   8  0.567971
31  20   8  0.542971
32   1   9  0.582518
33   5   9  0.603081
34  10   9  0.567960
35  20   9  0.543327
36   1  10  0.582518
37   5  10  0.611334
38  10  10  0.568912
39  20  10  0.546430
==================================================
Max auc score==> 0.6140860403295986
**************************************************
```

```
temp[2]==> 0     0.562490
1     0.603944
2     0.569142
3     0.522328
4     0.562490
5     0.608620
6     0.566346
7     0.517127
8     0.562490
9     0.600439
10    0.568027
11    0.514394
12    0.582518
13    0.612538
14    0.566836
15    0.515154
16    0.582518
17    0.614086
18    0.566288
19    0.514841
20    0.582518
21    0.611965
22    0.567515
23    0.522081
24    0.582518
25    0.612851
26    0.566654
27    0.529718
28    0.582518
29    0.612657
30    0.567971
31    0.542971
32    0.582518
33    0.603081
34    0.567960
35    0.543327
36    0.582518
37    0.611334
38    0.568912
39    0.546430
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
17  5  5  0.614086
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
     0   1         2
0    1   1  0.562490
1    5   1  0.603944
2   10   1  0.569142
3   20   1  0.522328
4    1   2  0.562490
5    5   2  0.608620
6   10   2  0.566346
7   20   2  0.517127
8    1   3  0.562490
9    5   3  0.600439
10  10   3  0.568027
11  20   3  0.514394
12   1   4  0.582518
13   5   4  0.612538
14  10   4  0.566836
15  20   4  0.515154
16   1   5  0.582518
17   5   5  0.614086
18  10   5  0.566288
19  20   5  0.514841
20   1   6  0.582518
21   5   6  0.611965
22  10   6  0.567515
23  20   6  0.522081
24   1   7  0.582518
25   5   7  0.612851
26  10   7  0.566654
27  20   7  0.529718
28   1   8  0.582518
```

```
28   1   8  0.582518
29   5   8  0.612657
30  10   8  0.567971
31  20   8  0.542971
32   1   9  0.582518
33   5   9  0.603081
34  10   9  0.567960
35  20   9  0.543327
36   1  10  0.582518
37   5  10  0.611334
38  10  10  0.568912
39  20  10  0.546430
==================================================
Max auc score==> 0.6140860403295986
**************************************************
temp[2]==> 0      0.562490
1      0.603944
2      0.569142
3      0.522328
4      0.562490
5      0.608620
6      0.566346
7      0.517127
8      0.562490
9      0.600439
10     0.568027
11     0.514394
12     0.582518
13     0.612538
14     0.566836
15     0.515154
16     0.582518
17     0.614086
18     0.566288
19     0.514841
20     0.582518
21     0.611965
22     0.567515
23     0.522081
24     0.582518
25     0.612851
26     0.566654
27     0.529718
28     0.582518
29     0.612657
30     0.567971
31     0.542971
32     0.582518
33     0.603081
34     0.567960
35     0.543327
36     0.582518
37     0.611334
38     0.568912
39     0.546430
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0   1          2
17   5   5   0.614086
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```



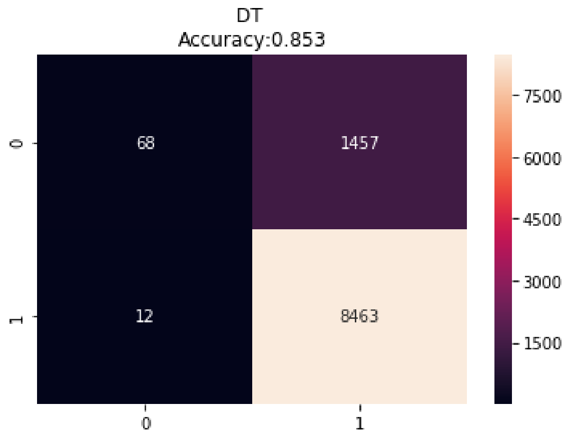


Plot Confusion Matrix

In [115]:

```
from sklearn.metrics import accuracy_score

#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for BOW")
cm =confusion_matrix(y_train, pred_bow_train, labels=None, sample_weight=None)
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT \nAccuracy:{0:.3f}'.format(accuracy_score(y_train, pred_bow_train)))
plt.show()
print("="*50)
print("Testing CM for BOW")
cm =confusion_matrix(y_test, pred_bow_test, labels=None, sample_weight=None)
summary.append(['BoW_GBDT',find_best_params(roc_auc_score_cv_bow_dict)['depth'],find_best_params(r
oc_auc_score_cv_bow_dict)['sample'],roc_auc_test])
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT  \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, pred_bow_test)))
plt.show()
```

```
Training CM for BOW
```

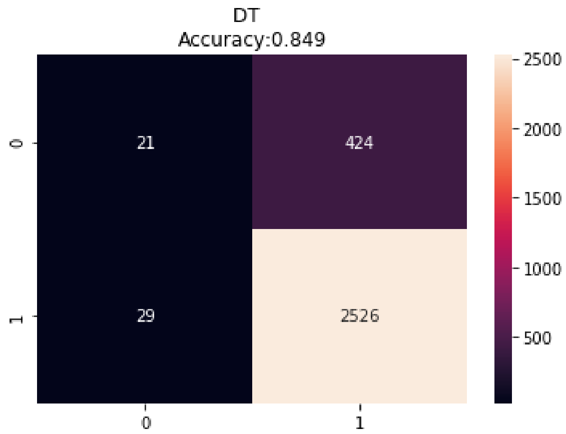



```
==================================================
Testing CM for BOW
     0  1         2
0    1  1  0.562490
1    5  1  0.603944
2   10  1  0.569142
3   20  1  0.522328
4    1  2  0.562490
5    5  2  0.608620
6   10  2  0.566346
7   20  2  0.517127
8    1  3  0.562490
9    5  3  0.600439
10  10  3  0.568027
11  20  3  0.514394
12   1  4  0.582518
13   5  4  0.612538
14  10  4  0.566836
15  20  4  0.515154
16   1  5  0.582518
17   5  5  0.614086
18  10  5  0.566288
19  20  5  0.514841
20   1  6  0.582518
21   5  6  0.611965
22  10  6  0.567515
23  20  6  0.522081
24   1  7  0.582518
```

```
25   5   7  0.612851
26  10   7  0.566654
27  20   7  0.529718
28   1   8  0.582518
29   5   8  0.612657
30  10   8  0.567971
31  20   8  0.542971
32   1   9  0.582518
33   5   9  0.603081
34  10   9  0.567960
35  20   9  0.543327
36   1  10  0.582518
37   5  10  0.611334
38  10  10  0.568912
39  20  10  0.546430
==================================================
Max auc score==> 0.6140860403295986
**************************************************
temp[2]==> 0     0.562490
1     0.603944
2     0.569142
3     0.522328
4     0.562490
5     0.608620
6     0.566346
7     0.517127
8     0.562490
9     0.600439
10    0.568027
11    0.514394
12    0.582518
13    0.612538
14    0.566836
15    0.515154
16    0.582518
17    0.614086
18    0.566288
19    0.514841
20    0.582518
21    0.611965
22    0.567515
23    0.522081
24    0.582518
25    0.612851
26    0.566654
27    0.529718
28    0.582518
29    0.612657
30    0.567971
31    0.542971
32    0.582518
33    0.603081
34    0.567960
35    0.543327
36    0.582518
37    0.611334
38    0.568912
39    0.546430
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
17  5  5  0.614086
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
     0   1         2
0    1   1  0.562490
1    5   1  0.603944
2   10   1  0.569142
3   20   1  0.522328
4    1   2  0.562490
5    5   2  0.608620
6   10   2  0.566346
7   20   2  0.517127
8    1   3  0.562490
9    5   3  0.600439
10  10   3  0.568027
```

```
11  20   3  0.514394
12   1   4  0.582518
13   5   4  0.612538
14  10   4  0.566836
15  20   4  0.515154
16   1   5  0.582518
17   5   5  0.614086
18  10   5  0.566288
19  20   5  0.514841
20   1   6  0.582518
21   5   6  0.611965
22  10   6  0.567515
23  20   6  0.522081
24   1   7  0.582518
25   5   7  0.612851
26  10   7  0.566654
27  20   7  0.529718
28   1   8  0.582518
29   5   8  0.612657
30  10   8  0.567971
31  20   8  0.542971
32   1   9  0.582518
33   5   9  0.603081
34  10   9  0.567960
35  20   9  0.543327
36   1  10  0.582518
37   5  10  0.611334
38  10  10  0.568912
39  20  10  0.546430
==================================================
Max auc score==> 0.6140860403295986
**************************************************
temp[2]==> 0     0.562490
1     0.603944
2     0.569142
3     0.522328
4     0.562490
5     0.608620
6     0.566346
7     0.517127
8     0.562490
9     0.600439
10    0.568027
11    0.514394
12    0.582518
13    0.612538
14    0.566836
15    0.515154
16    0.582518
17    0.614086
18    0.566288
19    0.514841
20    0.582518
21    0.611965
22    0.567515
23    0.522081
24    0.582518
25    0.612851
26    0.566654
27    0.529718
28    0.582518
29    0.612657
30    0.567971
31    0.542971
32    0.582518
33    0.603081
34    0.567960
35    0.543327
36    0.582518
37    0.611334
38    0.568912
39    0.546430
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
17  5  5  0.614086
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
```

```
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
####################################################
```



Apply GBDT on TFIDF SET 2

Train model for various values

In [116]:

```python
roc_auc_score_cv_tfidf_dict=[]
roc_auc_score_train_tfidf_dict=[]

num_trees = [1,2,3,4,5,6,7,8,9,10]
max_depth=[1, 5, 10, 20]


for n in tqdm(num_trees):
    for d in max_depth:
        #create instance of model
        dt=XGBClassifier(max_depth=d,n_estimators=n,learning_rate=0.01,nthread=1)

        #Fit the model on the training set
        dt.fit(X_train_tfidf,y_train)


        # predict the response on the crossvalidation train
        pred_tfidf_cv = dt.predict_proba(X_cv_tfidf)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_cv[:,1])

        #insert into dict
        roc_auc_score_cv_tfidf_dict.append([d,n,roc_auc_cv])

         # fitting the model on crossvalidation train
        dt.fit(X_train_tfidf, y_train)

        # predict the response on the train
        pred_tfidf_train = dt.predict_proba(X_train_tfidf)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_tfidf_train[:,1])

        #insert into dict
        roc_auc_score_train_tfidf_dict.append([d,n,roc_auc_train])

print(roc_auc_score_cv_tfidf_dict)
```

```
100%|██████████████████████████████████████████████████████████████████████| 10/10
[50:53<00:00, 406.50s/it]
```

```
[[1, 1, 0.5624898020232786], [5, 1, 0.5923286569944524], [10, 1, 0.5915485117752638], [20, 1,
0.5646772680300229], [1, 2, 0.5624898020232786], [5, 2, 0.5920881547101055], [10, 2,
0.5901360919993472], [20, 2, 0.5811342529642118], [1, 3, 0.5624898020232786], [5, 3
```

```
0.5921026018437943], [10, 3, 0.5905168164636136], [20, 3, 0.589778737898401], [1, 4,
0.5825177784727511], [5, 4, 0.5980535461764387], [10, 4, 0.5907003800445991], [20, 4,
0.5877433917110845], [1, 5, 0.5825177784727511], [5, 5, 0.5990512482323507], [10, 5,
0.6029375271946046], [20, 5, 0.5853506914228218], [1, 6, 0.5825177784727511], [5, 6,
0.597519852061351], [10, 6, 0.5852835547427391], [20, 6, 0.6104449377243555], [1, 7,
0.5825177784727511], [5, 7, 0.6059981949581204], [10, 7, 0.5885137638692484], [20, 7,
0.607510894838464], [1, 8, 0.5825177784727511], [5, 8, 0.606198755166975], [10, 8,
0.5877506152779288], [20, 8, 0.6072729420482976], [1, 9, 0.5825177784727511], [5, 9,
0.6064018648700099], [10, 9, 0.5867252937017295], [20, 9, 0.6053731439682367], [1, 10,
0.5825177784727511], [5, 10, 0.6047429939899924], [10, 10, 0.5867206196290655], [20, 10,
0.6073060854726422]]
```

3D Scatter Plot

In [117]:

```
x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_tfidf_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_tfidf_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross val')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'train')
data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(TFIDF)',scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='learning_rate'),
        zaxis = dict(title='AUC'),))

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

Find best Hper-parameter to train the model

In [118]:

```
find_best_params(roc_auc_score_cv_tfidf_dict)
```

```
     0   1         2
0    1   1  0.562490
1    5   1  0.592329
2   10   1  0.591549
3   20   1  0.564677
4    1   2  0.562490
5    5   2  0.592088
6   10   2  0.590136
7   20   2  0.581134
8    1   3  0.562490
9    5   3  0.592103
10  10   3  0.590517
11  20   3  0.589779
12   1   4  0.582518
13   5   4  0.598054
14  10   4  0.590700
15  20   4  0.587743
16   1   5  0.582518
17   5   5  0.599051
18  10   5  0.602938
19  20   5  0.585351
20   1   6  0.582518
21   5   6  0.597520
22  10   6  0.585284
23  20   6  0.610445
24   1   7  0.582518
25   5   7  0.605998
26  10   7  0.588514
27  20   7  0.607511
28   1   8  0.582518
29   5   8  0.606199
30  10   8  0.587751
31  20   8  0.607273
32   1   9  0.582518
33   5   9  0.606402
34  10   9  0.586725
35  20   9  0.605373
36   1  10  0.582518
37   5  10  0.604743
38  10  10  0.586721
39  20  10  0.607306
==================================================
Max auc score==> 0.6104449377243555
**************************************************
temp[2]==> 0     0.562490
1     0.592329
2     0.591549
3     0.564677
4     0.562490
5     0.592088
6     0.590136
7     0.581134
8     0.562490
9     0.592103
10    0.590517
11    0.589779
12    0.582518
13    0.598054
14    0.590700
15    0.587743
16    0.582518
17    0.599051
18    0.602938
19    0.585351
```

```
20    0.582518
21    0.597520
22    0.585284
23    0.610445
24    0.582518
25    0.605998
26    0.588514
27    0.607511
28    0.582518
29    0.606199
30    0.587751
31    0.607273
32    0.582518
33    0.606402
34    0.586725
35    0.605373
36    0.582518
37    0.604743
38    0.586721
39    0.607306
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
23  20  6  0.610445
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
```

Out[118]:

```
{'depth': 20, 'sample': 6}
```

Use best Hper-parameter to train the model

In [119]:

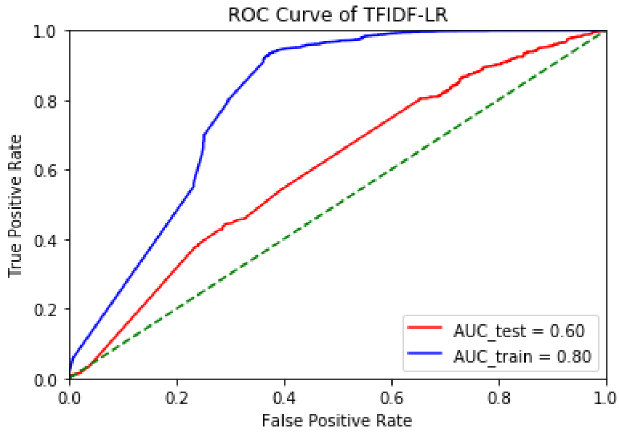
```
# train model on the best alpha
lr = XGBClassifier(learning_rate=0.01,max_depth=find_best_params(roc_auc_score_cv_tfidf_dict)
['depth'],n_estimators=find_best_params(roc_auc_score_cv_tfidf_dict)['sample'])

# fitting the model on crossvalidation train
lr.fit(X_train_tfidf, y_train)

# predict the response on the crossvalidation train
pred_tfidf_test = lr.predict(X_test_tfidf)
pred_tfidf_train = lr.predict(X_train_tfidf)

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_tfidf_test_scores=lr.predict_proba(X_test_tfidf)
pred_tfidf_train_scores=lr.predict_proba(X_train_tfidf)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_train_scores[:, 1])
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of TFIDF-LR')
plt.show()
```

```
    0  1         2
0   1  1  0.562490
1   5  1  0.592329
2  10  1  0.591549
3  20  1  0.564677
```

```
4     1   2  0.562490
5     5   2  0.592088
6    10   2  0.590136
7    20   2  0.581134
8     1   3  0.562490
9     5   3  0.592103
10   10   3  0.590517
11   20   3  0.589779
12    1   4  0.582518
13    5   4  0.598054
14   10   4  0.590700
15   20   4  0.587743
16    1   5  0.582518
17    5   5  0.599051
18   10   5  0.602938
19   20   5  0.585351
20    1   6  0.582518
21    5   6  0.597520
22   10   6  0.585284
23   20   6  0.610445
24    1   7  0.582518
25    5   7  0.605998
26   10   7  0.588514
27   20   7  0.607511
28    1   8  0.582518
29    5   8  0.606199
30   10   8  0.587751
31   20   8  0.607273
32    1   9  0.582518
33    5   9  0.606402
34   10   9  0.586725
35   20   9  0.605373
36    1  10  0.582518
37    5  10  0.604743
38   10  10  0.586721
39   20  10  0.607306
==================================================
Max auc score==> 0.6104449377243555
**************************************************
temp[2]==> 0      0.562490
1      0.592329
2      0.591549
3      0.564677
4      0.562490
5      0.592088
6      0.590136
7      0.581134
8      0.562490
9      0.592103
10     0.590517
11     0.589779
12     0.582518
13     0.598054
14     0.590700
15     0.587743
16     0.582518
17     0.599051
18     0.602938
19     0.585351
20     0.582518
21     0.597520
22     0.585284
23     0.610445
24     0.582518
25     0.605998
26     0.588514
27     0.607511
28     0.582518
29     0.606199
30     0.587751
31     0.607273
32     0.582518
33     0.606402
34     0.586725
35     0.605373
36     0.582518
37     0.604743
```

```
38     0.586721
39     0.607306
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
23  20  6  0.610445
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
     0   1         2
0    1   1  0.562490
1    5   1  0.592329
2   10   1  0.591549
3   20   1  0.564677
4    1   2  0.562490
5    5   2  0.592088
6   10   2  0.590136
7   20   2  0.581134
8    1   3  0.562490
9    5   3  0.592103
10  10   3  0.590517
11  20   3  0.589779
12   1   4  0.582518
13   5   4  0.598054
14  10   4  0.590700
15  20   4  0.587743
16   1   5  0.582518
17   5   5  0.599051
18  10   5  0.602938
19  20   5  0.585351
20   1   6  0.582518
21   5   6  0.597520
22  10   6  0.585284
23  20   6  0.610445
24   1   7  0.582518
25   5   7  0.605998
26  10   7  0.588514
27  20   7  0.607511
28   1   8  0.582518
29   5   8  0.606199
30  10   8  0.587751
31  20   8  0.607273
32   1   9  0.582518
33   5   9  0.606402
34  10   9  0.586725
35  20   9  0.605373
36   1  10  0.582518
37   5  10  0.604743
38  10  10  0.586721
39  20  10  0.607306
==================================================
Max auc score==> 0.6104449377243555
**************************************************
temp[2]==> 0     0.562490
1      0.592329
2      0.591549
3      0.564677
4      0.562490
5      0.592088
6      0.590136
7      0.581134
8      0.562490
9      0.592103
10     0.590517
11     0.589779
12     0.582518
13     0.598054
14     0.590700
15     0.587743
16     0.582518
17     0.599051
18     0.602938
19     0.585351
20     0.582518
21     0.597520
22     0.585284
23     0.610445
```

```
24    0.582518
25    0.605998
26    0.588514
27    0.607511
28    0.582518
29    0.606199
30    0.587751
31    0.607273
32    0.582518
33    0.606402
34    0.586725
35    0.605373
36    0.582518
37    0.604743
38    0.586721
39    0.607306
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
23  20  6  0.610445
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
```

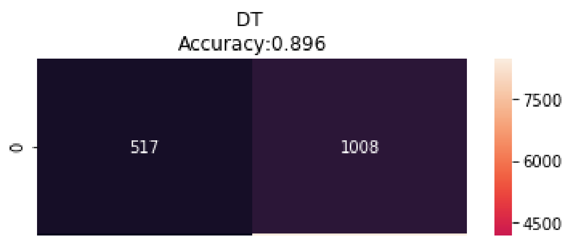


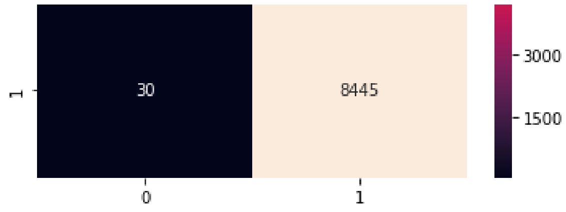
Plot Confusion Matrix

In [120]:

```
from sklearn.metrics import accuracy_score
#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for TFIDF")
cm =confusion_matrix(y_train, pred_tfidf_train, labels=None, sample_weight=None)
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT \nAccuracy:{0:.3f}'.format(accuracy_score(y_train, pred_tfidf_train)))
plt.show()
print("="*50)
print("Testing CM for TFIDF")
cm =confusion_matrix(y_test, pred_tfidf_test, labels=None, sample_weight=None)
summary.append(['Tfidf_GBDT',find_best_params(roc_auc_score_cv_tfidf_dict)
['depth'],find_best_params(roc_auc_score_cv_tfidf_dict)['sample'],roc_auc_test])
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT  \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, pred_tfidf_test)))
plt.show()
```

```
Training CM for TFIDF
```

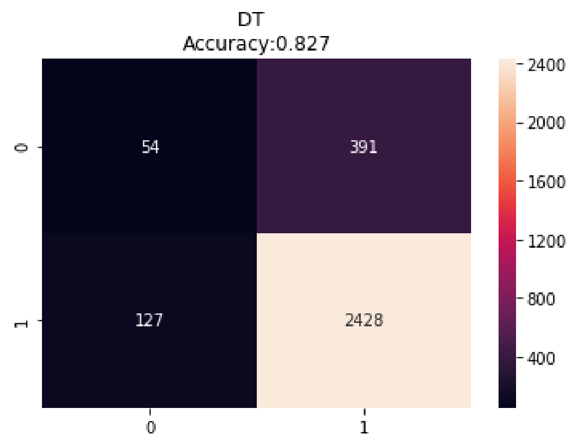





```
==================================================
Testing CM for TFIDF
     0   1         2
0    1   1  0.562490
1    5   1  0.592329
2   10   1  0.591549
3   20   1  0.564677
4    1   2  0.562490
5    5   2  0.592088
6   10   2  0.590136
7   20   2  0.581134
8    1   3  0.562490
9    5   3  0.592103
10  10   3  0.590517
11  20   3  0.589779
12   1   4  0.582518
13   5   4  0.598054
14  10   4  0.590700
15  20   4  0.587743
16   1   5  0.582518
17   5   5  0.599051
18  10   5  0.602938
19  20   5  0.585351
20   1   6  0.582518
21   5   6  0.597520
22  10   6  0.585284
23  20   6  0.610445
24   1   7  0.582518
25   5   7  0.605998
26  10   7  0.588514
27  20   7  0.607511
28   1   8  0.582518
29   5   8  0.606199
30  10   8  0.587751
31  20   8  0.607273
32   1   9  0.582518
33   5   9  0.606402
34  10   9  0.586725
35  20   9  0.605373
36   1  10  0.582518
37   5  10  0.604743
38  10  10  0.586721
39  20  10  0.607306
==================================================
Max auc score==> 0.6104449377243555
**************************************************
temp[2]==> 0     0.562490
1     0.592329
2     0.591549
3     0.564677
4     0.562490
5     0.592088
6     0.590136
7     0.581134
8     0.562490
9     0.592103
10    0.590517
11    0.589779
12    0.582518
13    0.598054
14    0.590700
15    0.587743
16    0.582518
17    0.599051
18    0.602938
19    0.585351
20    0.582518
21    0.597520
```

```
21      0.597520
22      0.585284
23      0.610445
24      0.582518
25      0.605998
26      0.588514
27      0.607511
28      0.582518
29      0.606199
30      0.587751
31      0.607273
32      0.582518
33      0.606402
34      0.586725
35      0.605373
36      0.582518
37      0.604743
38      0.586721
39      0.607306
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0  1         2
23  20  6  0.610445
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
     0   1         2
0    1   1  0.562490
1    5   1  0.592329
2   10   1  0.591549
3   20   1  0.564677
4    1   2  0.562490
5    5   2  0.592088
6   10   2  0.590136
7   20   2  0.581134
8    1   3  0.562490
9    5   3  0.592103
10  10   3  0.590517
11  20   3  0.589779
12   1   4  0.582518
13   5   4  0.598054
14  10   4  0.590700
15  20   4  0.587743
16   1   5  0.582518
17   5   5  0.599051
18  10   5  0.602938
19  20   5  0.585351
20   1   6  0.582518
21   5   6  0.597520
22  10   6  0.585284
23  20   6  0.610445
24   1   7  0.582518
25   5   7  0.605998
26  10   7  0.588514
27  20   7  0.607511
28   1   8  0.582518
29   5   8  0.606199
30  10   8  0.587751
31  20   8  0.607273
32   1   9  0.582518
33   5   9  0.606402
34  10   9  0.586725
35  20   9  0.605373
36   1  10  0.582518
37   5  10  0.604743
38  10  10  0.586721
39  20  10  0.607306
==================================================
Max auc score==> 0.6104449377243555
**************************************************
temp[2]==> 0      0.562490
1      0.592329
2      0.591549
3      0.564677
4      0.562490
5      0.592088
6      0.590136
7      0.581134
```

```
8      0.562490
9      0.592103
10     0.590517
11     0.589779
12     0.582518
13     0.598054
14     0.590700
15     0.587743
16     0.582518
17     0.599051
18     0.602938
19     0.585351
20     0.582518
21     0.597520
22     0.585284
23     0.610445
24     0.582518
25     0.605998
26     0.588514
27     0.607511
28     0.582518
29     0.606199
30     0.587751
31     0.607273
32     0.582518
33     0.606402
34     0.586725
35     0.605373
36     0.582518
37     0.604743
38     0.586721
39     0.607306
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>        0   1          2
23   20   6   0.610445
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 20.0
##################################################
```



Apply GBDT on W2V SET 3

Train model for various values

In [121]:

```python
roc_auc_score_cv_w2v_dict=[]
roc_auc_score_train_w2v_dict=[]

num_trees = [1,2,3,4,5,6,7,8,9,10]
max_depth=[1, 5, 10, 20]


for n in tqdm(num_trees):
    for d in max_depth:
```

```
    for d in max_depth:
        #create instance of model
        dt=XGBClassifier(max_depth=d,n_estimators=n,learning_rate=0.01,nthread=1)

        #Fit the model on the training set
        dt.fit(X_train_w2v,y_train)


        # predict the response on the crossvalidation train
        pred_w2v_cv = dt.predict_proba(X_cv_w2v)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_w2v_cv[:,1])

        #insert into dict
        roc_auc_score_cv_w2v_dict.append([d,n,roc_auc_cv])

         # fitting the model on crossvalidation train
        dt.fit(X_train_w2v, y_train)

        # predict the response on the train
        pred_w2v_train = dt.predict_proba(X_train_w2v)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_w2v_train[:,1])

        #insert into dict
        roc_auc_score_train_w2v_dict.append([d,n,roc_auc_train])

print(roc_auc_score_cv_w2v_dict)
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 10/10
[11:17<00:00, 91.91s/it]
```

```
[[1, 1, 0.5624898020232786], [5, 1, 0.6081924596160122], [10, 1, 0.5684441456814969], [20, 1,
0.5562749850429676], [1, 2, 0.5624898020232786], [5, 2, 0.6246927859512672], [10, 2,
0.5929753786848689], [20, 2, 0.5711194149080823], [1, 3, 0.5624898020232786], [5, 3,
0.6223451267268574], [10, 3, 0.5866207644403351], [20, 3, 0.5964431156858478], [1, 4,
0.5825177784727511], [5, 4, 0.6256301499782443], [10, 4, 0.5901887815457414], [20, 4,
0.5916895837865768], [1, 5, 0.5825177784727511], [5, 5, 0.6412997661264005], [10, 5,
0.5903617222343087], [20, 5, 0.5987142900848472], [1, 6, 0.5825177784727511], [5, 6,
0.6388140093005548], [10, 6, 0.6172792817904929], [20, 6, 0.6035681020885456], [1, 7,
0.5825177784727511], [5, 7, 0.6425507179375611], [10, 7, 0.6162939022897858], [20, 7,
0.6036768805069075], [1, 8, 0.5825177784727511], [5, 8, 0.6409109682638965], [10, 8,
0.6168794361198738], [20, 8, 0.6180471044544763], [1, 9, 0.5825177784727511], [5, 9,
0.6453734329109104], [10, 9, 0.623746498694659], [20, 9, 0.6176944244261938], [1, 10,
0.5825177784727511], [5, 10, 0.6453594106929186], [10, 10, 0.6246549684542587], [20, 10,
0.617846119329925]]
```

3D Scatter Plot

In [122]:

```
x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_w2v_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_w2v_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross val')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'train')
```

```
data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(W2V)',scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='learning_rate'),
        zaxis = dict(title='AUC'),))

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

Find best Hper-parameter to train the model

In [123]:

```
find_best_params(roc_auc_score_cv_w2v_dict)
```

```
     0  1         2
0    1  1  0.562490
1    5  1  0.608192
2   10  1  0.568444
3   20  1  0.556275
4    1  2  0.562490
5    5  2  0.624693
6   10  2  0.592975
7   20  2  0.571119
8    1  3  0.562490
9    5  3  0.622345
10  10  3  0.586621
11  20  3  0.596443
12   1  4  0.582518
13   5  4  0.625630
14  10  4  0.590189
15  20  4  0.591690
16   1  5  0.582518
17   5  5  0.641300
18  10  5  0.590362
19  20  5  0.598714
20   1  6  0.582518
21   5  6  0.638814
22  10  6  0.617279
23  20  6  0.603568
24   1  7  0.582518
25   5  7  0.642551
```

```
25   5   7  0.642551
26  10   7  0.616294
27  20   7  0.603677
28   1   8  0.582518
29   5   8  0.640911
30  10   8  0.616879
31  20   8  0.618047
32   1   9  0.582518
33   5   9  0.645373
34  10   9  0.623746
35  20   9  0.617694
36   1  10  0.582518
37   5  10  0.645359
38  10  10  0.624655
39  20  10  0.617846
==================================================
Max auc score==> 0.6453734329109104
**************************************************
temp[2]==> 0     0.562490
1     0.608192
2     0.568444
3     0.556275
4     0.562490
5     0.624693
6     0.592975
7     0.571119
8     0.562490
9     0.622345
10    0.586621
11    0.596443
12    0.582518
13    0.625630
14    0.590189
15    0.591690
16    0.582518
17    0.641300
18    0.590362
19    0.598714
20    0.582518
21    0.638814
22    0.617279
23    0.603568
24    0.582518
25    0.642551
26    0.616294
27    0.603677
28    0.582518
29    0.640911
30    0.616879
31    0.618047
32    0.582518
33    0.645373
34    0.623746
35    0.617694
36    0.582518
37    0.645359
38    0.624655
39    0.617846
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
33  5  9  0.645373
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```

Out[123]:

```
{'depth': 5, 'sample': 9}
```

Use best Hper-parameter to train the model

In [124]:

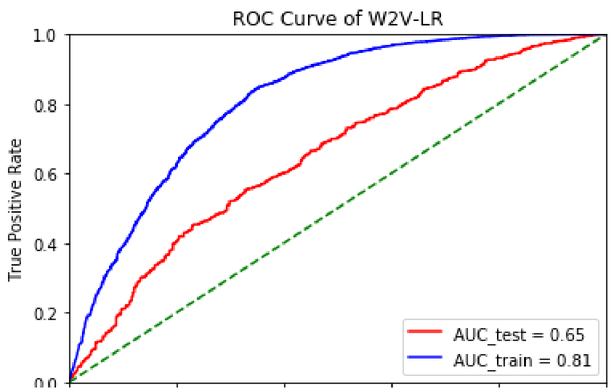
```
from xgboost import XGBClassifier
```

```
# train model on the best alpha
lr = XGBClassifier(learning_rate=0.1,max_depth=find_best_params(roc_auc_score_cv_w2v_dict)['depth'
],n_estimators=find_best_params(roc_auc_score_cv_w2v_dict)['sample'])

# fitting the model on crossvalidation train
lr.fit(X_train_w2v, y_train)

# predict the response on the crossvalidation train
pred_w2v_test = lr.predict(X_test_w2v)
pred_w2v_train = lr.predict(X_train_w2v)

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_w2v_test_scores=lr.predict_proba(X_test_w2v)
pred_w2v_train_scores=lr.predict_proba(X_train_w2v)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_w2v_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_w2v_train_scores[:, 1])
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of W2V-LR')
plt.show()
```

```
     0   1         2
0    1   1  0.562490
1    5   1  0.608192
2   10   1  0.568444
3   20   1  0.556275
4    1   2  0.562490
5    5   2  0.624693
6   10   2  0.592975
7   20   2  0.571119
8    1   3  0.562490
9    5   3  0.622345
10  10   3  0.586621
11  20   3  0.596443
12   1   4  0.582518
13   5   4  0.625630
14  10   4  0.590189
15  20   4  0.591690
16   1   5  0.582518
17   5   5  0.641300
18  10   5  0.590362
19  20   5  0.598714
20   1   6  0.582518
21   5   6  0.638814
22  10   6  0.617279
23  20   6  0.603568
24   1   7  0.582518
25   5   7  0.642551
26  10   7  0.616294
27  20   7  0.603677
28   1   8  0.582518
29   5   8  0.640911
30  10   8  0.616879
31  20   8  0.618047
32   1   9  0.582518
33   5   9  0.645373
34  10   9  0.623746
35  20   9  0.617694
36   1  10  0.582518
37   5  10  0.645359
38  10  10  0.624655
39  20  10  0.617846
==================================================
Max auc score==> 0.6453734329109104
**************************************************
```

```
temp[2]==> 0     0.562490
1     0.608192
2     0.568444
3     0.556275
4     0.562490
5     0.624693
6     0.592975
7     0.571119
8     0.562490
9     0.622345
10    0.586621
11    0.596443
12    0.582518
13    0.625630
14    0.590189
15    0.591690
16    0.582518
17    0.641300
18    0.590362
19    0.598714
20    0.582518
21    0.638814
22    0.617279
23    0.603568
24    0.582518
25    0.642551
26    0.616294
27    0.603677
28    0.582518
29    0.640911
30    0.616879
31    0.618047
32    0.582518
33    0.645373
34    0.623746
35    0.617694
36    0.582518
37    0.645359
38    0.624655
39    0.617846
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
33  5  9  0.645373
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
     0   1         2
0    1   1  0.562490
1    5   1  0.608192
2   10   1  0.568444
3   20   1  0.556275
4    1   2  0.562490
5    5   2  0.624693
6   10   2  0.592975
7   20   2  0.571119
8    1   3  0.562490
9    5   3  0.622345
10  10   3  0.586621
11  20   3  0.596443
12   1   4  0.582518
13   5   4  0.625630
14  10   4  0.590189
15  20   4  0.591690
16   1   5  0.582518
17   5   5  0.641300
18  10   5  0.590362
19  20   5  0.598714
20   1   6  0.582518
21   5   6  0.638814
22  10   6  0.617279
23  20   6  0.603568
24   1   7  0.582518
25   5   7  0.642551
26  10   7  0.616294
27  20   7  0.603677
28   1   8  0.582518
```

```
29   5   8  0.640911
30  10   8  0.616879
31  20   8  0.618047
32   1   9  0.582518
33   5   9  0.645373
34  10   9  0.623746
35  20   9  0.617694
36   1  10  0.582518
37   5  10  0.645359
38  10  10  0.624655
39  20  10  0.617846
==================================================
Max auc score==> 0.6453734329109104
**************************************************
temp[2]==> 0     0.562490
1      0.608192
2      0.568444
3      0.556275
4      0.562490
5      0.624693
6      0.592975
7      0.571119
8      0.562490
9      0.622345
10     0.586621
11     0.596443
12     0.582518
13     0.625630
14     0.590189
15     0.591690
16     0.582518
17     0.641300
18     0.590362
19     0.598714
20     0.582518
21     0.638814
22     0.617279
23     0.603568
24     0.582518
25     0.642551
26     0.616294
27     0.603677
28     0.582518
29     0.640911
30     0.616879
31     0.618047
32     0.582518
33     0.645373
34     0.623746
35     0.617694
36     0.582518
37     0.645359
38     0.624655
39     0.617846
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
33  5  9  0.645373
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```

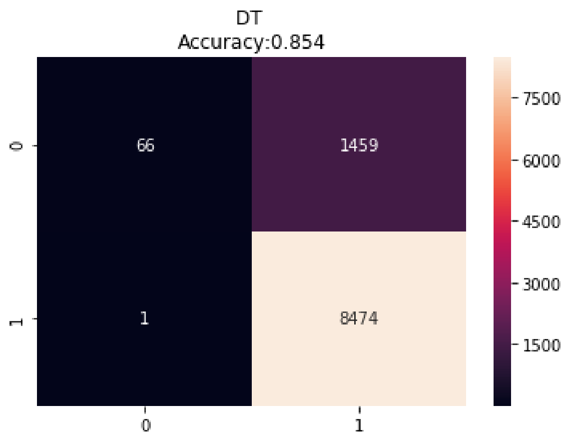


Plot Confusion Matrix

In [125]:

```
from sklearn.metrics import accuracy_score

from sklearn.metrics import accuracy_score

#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for W2V")
cm =confusion_matrix(y_train, pred_w2v_train, labels=None, sample_weight=None)
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT \nAccuracy:{0:.3f}'.format(accuracy_score(y_train, pred_w2v_train)))
plt.show()
print("="*50)
print("Testing CM for W2V")
cm =confusion_matrix(y_test, pred_w2v_test, labels=None, sample_weight=None)
summary.append(['W2v_GBDT',find_best_params(roc_auc_score_cv_w2v_dict)['depth'],find_best_params(r
oc_auc_score_cv_w2v_dict)['sample'],roc_auc_test])
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT  \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, pred_w2v_test)))
plt.show()
```

```
Training CM for W2V
```

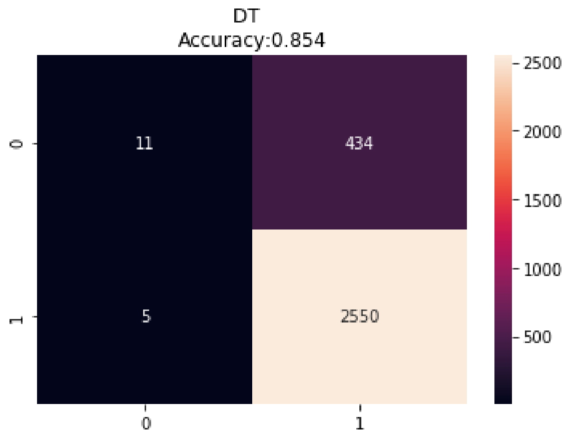



```
==================================================
Testing CM for W2V
     0   1         2
0    1   1  0.562490
1    5   1  0.608192
2   10   1  0.568444
3   20   1  0.556275
4    1   2  0.562490
5    5   2  0.624693
6   10   2  0.592975
7   20   2  0.571119
8    1   3  0.562490
9    5   3  0.622345
10  10   3  0.586621
11  20   3  0.596443
12   1   4  0.582518
13   5   4  0.625630
14  10   4  0.590189
15  20   4  0.591690
16   1   5  0.582518
17   5   5  0.641300
18  10   5  0.590362
19  20   5  0.598714
20   1   6  0.582518
21   5   6  0.638814
22  10   6  0.617279
```

```
23  20   6  0.603568
24   1   7  0.582518
25   5   7  0.642551
26  10   7  0.616294
27  20   7  0.603677
28   1   8  0.582518
29   5   8  0.640911
30  10   8  0.616879
31  20   8  0.618047
32   1   9  0.582518
33   5   9  0.645373
34  10   9  0.623746
35  20   9  0.617694
36   1  10  0.582518
37   5  10  0.645359
38  10  10  0.624655
39  20  10  0.617846
==================================================
Max auc score==> 0.6453734329109104
**************************************************
temp[2]==> 0     0.562490
1     0.608192
2     0.568444
3     0.556275
4     0.562490
5     0.624693
6     0.592975
7     0.571119
8     0.562490
9     0.622345
10    0.586621
11    0.596443
12    0.582518
13    0.625630
14    0.590189
15    0.591690
16    0.582518
17    0.641300
18    0.590362
19    0.598714
20    0.582518
21    0.638814
22    0.617279
23    0.603568
24    0.582518
25    0.642551
26    0.616294
27    0.603677
28    0.582518
29    0.640911
30    0.616879
31    0.618047
32    0.582518
33    0.645373
34    0.623746
35    0.617694
36    0.582518
37    0.645359
38    0.624655
39    0.617846
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
33  5  9  0.645373
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
     0   1         2
0    1   1  0.562490
1    5   1  0.608192
2   10   1  0.568444
3   20   1  0.556275
4    1   2  0.562490
5    5   2  0.624693
6   10   2  0.592975
7   20   2  0.571119
8    1   3  0.562490
```

```
9    5   3  0.622345
10  10   3  0.586621
11  20   3  0.596443
12   1   4  0.582518
13   5   4  0.625630
14  10   4  0.590189
15  20   4  0.591690
16   1   5  0.582518
17   5   5  0.641300
18  10   5  0.590362
19  20   5  0.598714
20   1   6  0.582518
21   5   6  0.638814
22  10   6  0.617279
23  20   6  0.603568
24   1   7  0.582518
25   5   7  0.642551
26  10   7  0.616294
27  20   7  0.603677
28   1   8  0.582518
29   5   8  0.640911
30  10   8  0.616879
31  20   8  0.618047
32   1   9  0.582518
33   5   9  0.645373
34  10   9  0.623746
35  20   9  0.617694
36   1  10  0.582518
37   5  10  0.645359
38  10  10  0.624655
39  20  10  0.617846
==================================================
Max auc score==> 0.6453734329109104
**************************************************
temp[2]==> 0     0.562490
1     0.608192
2     0.568444
3     0.556275
4     0.562490
5     0.624693
6     0.592975
7     0.571119
8     0.562490
9     0.622345
10    0.586621
11    0.596443
12    0.582518
13    0.625630
14    0.590189
15    0.591690
16    0.582518
17    0.641300
18    0.590362
19    0.598714
20    0.582518
21    0.638814
22    0.617279
23    0.603568
24    0.582518
25    0.642551
26    0.616294
27    0.603677
28    0.582518
29    0.640911
30    0.616879
31    0.618047
32    0.582518
33    0.645373
34    0.623746
35    0.617694
36    0.582518
37    0.645359
38    0.624655
39    0.617846
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0  1         2
33  5  9  0.645373
```

```
33   5   9   0.645373
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
####################################################
```



Apply GBDT on TFIDFW2V SET 4

Train model for various values

In [126]:

```
roc_auc_score_cv_tfidf_w2v_dict=[]
roc_auc_score_train_tfidf_w2v_dict=[]

num_trees = [1,2,3,4,5,6,7,8,9,10]
max_depth=[1, 5, 10, 20]


for n in tqdm(num_trees):
    for d in max_depth:
        #create instance of model
        dt=XGBClassifier(max_depth=d,n_estimators=n,learning_rate=0.01,nthread=1)

        #Fit the model on the training set
        dt.fit(X_train_tfidf_w2v,y_train)


        # predict the response on the crossvalidation train
        pred_tfidf_w2v_cv = dt.predict_proba(X_cv_tfidf_w2v)

        #evaluate CV roc_auc
        roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_w2v_cv[:,1])

        #insert into dict
        roc_auc_score_cv_tfidf_w2v_dict.append([d,n,roc_auc_cv])

         # fitting the model on crossvalidation train
        dt.fit(X_train_tfidf_w2v, y_train)

        # predict the response on the train
        pred_tfidf_w2v_train = dt.predict_proba(X_train_tfidf_w2v)

        #evaluate train roc_auc
        roc_auc_train =roc_auc_score(y_train,pred_tfidf_w2v_train[:,1])

        #insert into dict
        roc_auc_score_train_tfidf_w2v_dict.append([d,n,roc_auc_train])

print(roc_auc_score_cv_tfidf_w2v_dict)
```

```
100%|████████████████████████████████████████████████████████████████████████| 10/10
[11:19<00:00, 92.25s/it]
```

```
[[1, 1, 0.5624898020232786], [5, 1, 0.6252812941912325], [10, 1, 0.5844651671108452], [20, 1,
```

```
0.5548481181333623], [1, 2, 0.5624898020232786], [5, 2, 0.6213805680952899], [10, 2,
0.5767372253344937], [20, 2, 0.5566914024257588], [1, 3, 0.5624898020232786], [5, 3,
0.6230309406613728], [10, 3, 0.5807777486946589], [20, 3, 0.5651854672033069], [1, 4,
0.5825177784727511], [5, 4, 0.628655124823235], [10, 4, 0.5807853971772001], [20, 4,
0.5686719004949418], [1, 5, 0.5825177784727511], [5, 5, 0.6314400222995757], [10, 5,
0.5848458915751116], [20, 5, 0.5725042151637115], [1, 6, 0.5825177784727511], [5, 6,
0.6437404818883933], [10, 6, 0.587624840231698], [20, 6, 0.5747932360219732], [1, 7,
0.5825177784727511], [5, 7, 0.6413307849722615], [10, 7, 0.593495900413358], [20, 7,
0.5724311296638747], [1, 8, 0.5825177784727511], [5, 8, 0.646578068911128], [10, 8,
0.5985948887740673], [20, 8, 0.5757424976884586], [1, 9, 0.5825177784727511], [5, 9,
0.6468215456053519], [10, 9, 0.6117358316110084], [20, 9, 0.5759902235396497], [1, 10,
0.5825177784727511], [5, 10, 0.6487761578102904], [10, 10, 0.6110410944468618], [20, 10,
0.5793946481018165]]
```

3D Scatter Plot

In [127]:

```
x1=[]
y1=[]
z1=[]

x2=[]
y2=[]
z2=[]

for value in roc_auc_score_cv_tfidf_w2v_dict:
    x1.append(value[0])
    y1.append(value[1])
    z1.append(value[2])

for value in roc_auc_score_train_tfidf_w2v_dict:
    x2.append(value[0])
    y2.append(value[1])
    z2.append(value[2])

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=x1,y=y1,z=z1, name = 'Cross val')
trace2 = go.Scatter3d(x=x2,y=y2,z=z2, name = 'train')
data = [trace1, trace2]

layout = go.Layout(title='Depth vs split size vs AUC(TFIDF_W2V)',scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='learning_rate'),
        zaxis = dict(title='AUC'),))

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

Find best Hper-parameter to train the model

In [128]:

```
find_best_params(roc_auc_score_cv_tfidf_w2v_dict)
```

```
     0   1         2
0    1   1  0.562490
1    5   1  0.625281
2   10   1  0.584465
3   20   1  0.554848
4    1   2  0.562490
5    5   2  0.621381
6   10   2  0.576737
7   20   2  0.556691
8    1   3  0.562490
9    5   3  0.623031
10  10   3  0.580778
11  20   3  0.565185
12   1   4  0.582518
13   5   4  0.628655
14  10   4  0.580785
15  20   4  0.568672
16   1   5  0.582518
17   5   5  0.631440
18  10   5  0.584846
19  20   5  0.572504
20   1   6  0.582518
21   5   6  0.643740
22  10   6  0.587625
23  20   6  0.574793
24   1   7  0.582518
25   5   7  0.641331
26  10   7  0.593496
27  20   7  0.572431
28   1   8  0.582518
29   5   8  0.646578
30  10   8  0.598595
31  20   8  0.575742
32   1   9  0.582518
33   5   9  0.646822
34  10   9  0.611736
35  20   9  0.575990
36   1  10  0.582518
37   5  10  0.648776
38  10  10  0.611041
39  20  10  0.579395
==================================================
Max auc score==> 0.6487761578102904
**************************************************
temp[2]==> 0     0.562490
1     0.625281
2     0.584465
3     0.554848
4     0.562490
5     0.621381
6     0.576737
7     0.556691
8     0.562490
9     0.623031
10    0.580778
11    0.565185
12    0.582518
13    0.628655
14    0.580785
15    0.568672
16    0.582518
17    0.631440
```

```
18    0.584846
19    0.572504
20    0.582518
21    0.643740
22    0.587625
23    0.574793
24    0.582518
25    0.641331
26    0.593496
27    0.572431
28    0.582518
29    0.646578
30    0.598595
31    0.575742
32    0.582518
33    0.646822
34    0.611736
35    0.575990
36    0.582518
37    0.648776
38    0.611041
39    0.579395
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0   1         2
37  5  10  0.648776
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
###################################################
```

Out[128]:

```
{'depth': 5, 'sample': 10}
```

Use best Hper-parameter to train the model

In [129]:

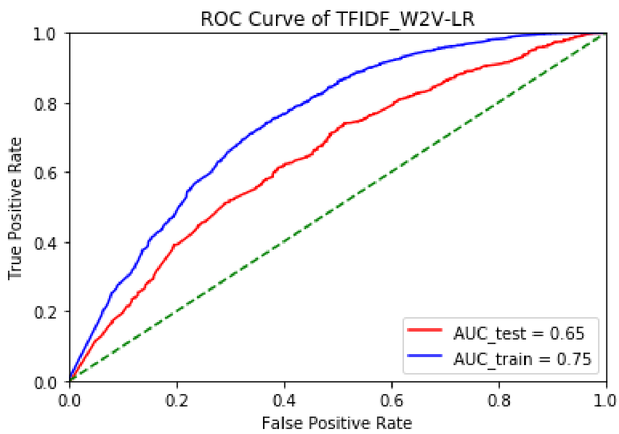
```
# train model on the best alpha
lr = XGBClassifier(learning_rate=0.01,max_depth=find_best_params(roc_auc_score_cv_tfidf_w2v_dict)[
'depth'],n_estimators=find_best_params(roc_auc_score_cv_tfidf_w2v_dict)['sample'])

# fitting the model on crossvalidation train
lr.fit(X_train_tfidf_w2v, y_train)

# predict the response on the crossvalidation train
pred_tfidf_w2v_test = lr.predict(X_test_tfidf_w2v)
pred_tfidf_w2v_train = lr.predict(X_train_tfidf_w2v)

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_tfidf_w2v_test_scores=lr.predict_proba(X_test_tfidf_w2v)
pred_tfidf_w2v_train_scores=lr.predict_proba(X_train_tfidf_w2v)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_w2v_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_w2v_train_scores[:, 1])
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)
plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of TFIDF_W2V-LR')
plt.show()
```

```
     0   1         2
0    1   1  0.562490
1    5   1  0.625281
2   10   1  0.584465
```

```
3   20   1  0.554848
4    1   2  0.562490
5    5   2  0.621381
6   10   2  0.576737
7   20   2  0.556691
8    1   3  0.562490
9    5   3  0.623031
10  10   3  0.580778
11  20   3  0.565185
12   1   4  0.582518
13   5   4  0.628655
14  10   4  0.580785
15  20   4  0.568672
16   1   5  0.582518
17   5   5  0.631440
18  10   5  0.584846
19  20   5  0.572504
20   1   6  0.582518
21   5   6  0.643740
22  10   6  0.587625
23  20   6  0.574793
24   1   7  0.582518
25   5   7  0.641331
26  10   7  0.593496
27  20   7  0.572431
28   1   8  0.582518
29   5   8  0.646578
30  10   8  0.598595
31  20   8  0.575742
32   1   9  0.582518
33   5   9  0.646822
34  10   9  0.611736
35  20   9  0.575990
36   1  10  0.582518
37   5  10  0.648776
38  10  10  0.611041
39  20  10  0.579395
==================================================
Max auc score==> 0.6487761578102904
**************************************************
temp[2]==> 0     0.562490
1     0.625281
2     0.584465
3     0.554848
4     0.562490
5     0.621381
6     0.576737
7     0.556691
8     0.562490
9     0.623031
10    0.580778
11    0.565185
12    0.582518
13    0.628655
14    0.580785
15    0.568672
16    0.582518
17    0.631440
18    0.584846
19    0.572504
20    0.582518
21    0.643740
22    0.587625
23    0.574793
24    0.582518
25    0.641331
26    0.593496
27    0.572431
28    0.582518
29    0.646578
30    0.598595
31    0.575742
32    0.582518
33    0.646822
34    0.611736
35    0.575990
36    0.582518
```

```
37    0.648776
38    0.611041
39    0.579395
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0   1         2
37  5  10  0.648776
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
     0   1         2
0    1   1  0.562490
1    5   1  0.625281
2   10   1  0.584465
3   20   1  0.554848
4    1   2  0.562490
5    5   2  0.621381
6   10   2  0.576737
7   20   2  0.556691
8    1   3  0.562490
9    5   3  0.623031
10  10   3  0.580778
11  20   3  0.565185
12   1   4  0.582518
13   5   4  0.628655
14  10   4  0.580785
15  20   4  0.568672
16   1   5  0.582518
17   5   5  0.631440
18  10   5  0.584846
19  20   5  0.572504
20   1   6  0.582518
21   5   6  0.643740
22  10   6  0.587625
23  20   6  0.574793
24   1   7  0.582518
25   5   7  0.641331
26  10   7  0.593496
27  20   7  0.572431
28   1   8  0.582518
29   5   8  0.646578
30  10   8  0.598595
31  20   8  0.575742
32   1   9  0.582518
33   5   9  0.646822
34  10   9  0.611736
35  20   9  0.575990
36   1  10  0.582518
37   5  10  0.648776
38  10  10  0.611041
39  20  10  0.579395
==================================================
Max auc score==> 0.6487761578102904
**************************************************
temp[2]==> 0     0.562490
1     0.625281
2     0.584465
3     0.554848
4     0.562490
5     0.621381
6     0.576737
7     0.556691
8     0.562490
9     0.623031
10    0.580778
11    0.565185
12    0.582518
13    0.628655
14    0.580785
15    0.568672
16    0.582518
17    0.631440
18    0.584846
19    0.572504
20    0.582518
21    0.643740
22    0.587625
```

```
23     0.574793
24     0.582518
25     0.641331
26     0.593496
27     0.572431
28     0.582518
29     0.646578
30     0.598595
31     0.575742
32     0.582518
33     0.646822
34     0.611736
35     0.575990
36     0.582518
37     0.648776
38     0.611041
39     0.579395
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>      0    1          2
37   5   10   0.648776
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```

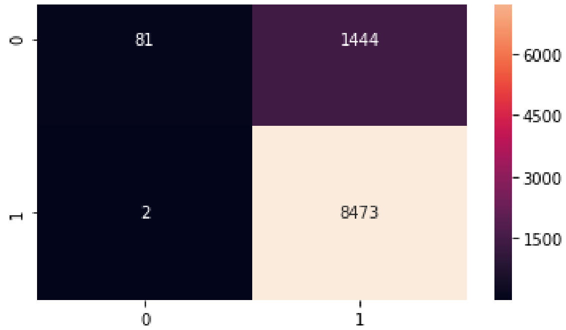


Plot Confusion Matrix

In [130]:

```python
from sklearn.metrics import accuracy_score

from sklearn.metrics import accuracy_score

#https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
from sklearn.metrics import confusion_matrix
print("Training CM for TFIDF_W2V")
cm =confusion_matrix(y_train, pred_tfidf_w2v_train, labels=None, sample_weight=None)
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT \nAccuracy:{0:.3f}'.format(accuracy_score(y_train, pred_tfidf_w2v_train)))
plt.show()
print("="*50)
print("Testing CM for TFIDF_W2V")
cm =confusion_matrix(y_test, pred_tfidf_w2v_test, labels=None, sample_weight=None)
summary.append(['Tfidf_w2v_GBDT',find_best_params(roc_auc_score_cv_tfidf_w2v_dict)
['depth'],find_best_params(roc_auc_score_cv_tfidf_w2v_dict)['sample'],roc_auc_test])
sns.heatmap(cm, annot=True,fmt="d")
plt.title('DT  \nAccuracy:{0:.3f}'.format(accuracy_score(y_test, pred_tfidf_w2v_test)))
plt.show()
```

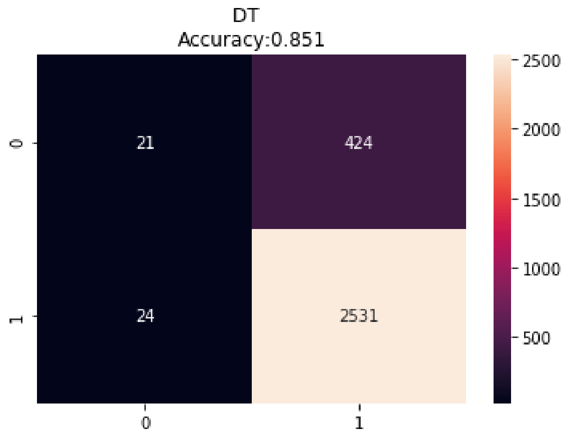
```
Training CM for TFIDF_W2V
```




```
==================================================
Testing CM for TFIDF_W2V
     0   1         2
0    1   1  0.562490
1    5   1  0.625281
2   10   1  0.584465
3   20   1  0.554848
4    1   2  0.562490
5    5   2  0.621381
6   10   2  0.576737
7   20   2  0.556691
8    1   3  0.562490
9    5   3  0.623031
10  10   3  0.580778
11  20   3  0.565185
12   1   4  0.582518
13   5   4  0.628655
14  10   4  0.580785
15  20   4  0.568672
16   1   5  0.582518
17   5   5  0.631440
18  10   5  0.584846
19  20   5  0.572504
20   1   6  0.582518
21   5   6  0.643740
22  10   6  0.587625
23  20   6  0.574793
24   1   7  0.582518
25   5   7  0.641331
26  10   7  0.593496
27  20   7  0.572431
28   1   8  0.582518
29   5   8  0.646578
30  10   8  0.598595
31  20   8  0.575742
32   1   9  0.582518
33   5   9  0.646822
34  10   9  0.611736
35  20   9  0.575990
36   1  10  0.582518
37   5  10  0.648776
38  10  10  0.611041
39  20  10  0.579395
==================================================
Max auc score==> 0.6487761578102904
**************************************************
temp[2]==> 0     0.562490
1     0.625281
2     0.584465
3     0.554848
4     0.562490
5     0.621381
6     0.576737
7     0.556691
8     0.562490
9     0.623031
10    0.580778
11    0.565185
12    0.582518
13    0.628655
14    0.580785
15    0.568672
16    0.582518
```

```
17    0.631440
18    0.584846
19    0.572504
20    0.582518
21    0.643740
22    0.587625
23    0.574793
24    0.582518
25    0.641331
26    0.593496
27    0.572431
28    0.582518
29    0.646578
30    0.598595
31    0.575742
32    0.582518
33    0.646822
34    0.611736
35    0.575990
36    0.582518
37    0.648776
38    0.611041
39    0.579395
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0   1         2
37  5  10  0.648776
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
     0   1         2
0    1   1  0.562490
1    5   1  0.625281
2   10   1  0.584465
3   20   1  0.554848
4    1   2  0.562490
5    5   2  0.621381
6   10   2  0.576737
7   20   2  0.556691
8    1   3  0.562490
9    5   3  0.623031
10  10   3  0.580778
11  20   3  0.565185
12   1   4  0.582518
13   5   4  0.628655
14  10   4  0.580785
15  20   4  0.568672
16   1   5  0.582518
17   5   5  0.631440
18  10   5  0.584846
19  20   5  0.572504
20   1   6  0.582518
21   5   6  0.643740
22  10   6  0.587625
23  20   6  0.574793
24   1   7  0.582518
25   5   7  0.641331
26  10   7  0.593496
27  20   7  0.572431
28   1   8  0.582518
29   5   8  0.646578
30  10   8  0.598595
31  20   8  0.575742
32   1   9  0.582518
33   5   9  0.646822
34  10   9  0.611736
35  20   9  0.575990
36   1  10  0.582518
37   5  10  0.648776
38  10  10  0.611041
39  20  10  0.579395
==================================================
Max auc score==> 0.6487761578102904
**************************************************
temp[2]==> 0     0.562490
1     0.625281
2     0.584465
```

```
3      0.554848
4      0.562490
5      0.621381
6      0.576737
7      0.556691
8      0.562490
9      0.623031
10     0.580778
11     0.565185
12     0.582518
13     0.628655
14     0.580785
15     0.568672
16     0.582518
17     0.631440
18     0.584846
19     0.572504
20     0.582518
21     0.643740
22     0.587625
23     0.574793
24     0.582518
25     0.641331
26     0.593496
27     0.572431
28     0.582518
29     0.646578
30     0.598595
31     0.575742
32     0.582518
33     0.646822
34     0.611736
35     0.575990
36     0.582518
37     0.648776
38     0.611041
39     0.579395
Name: 2, dtype: float64
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
temp[temp[2]==max(temp[2])]==>     0   1         2
37  5  10  0.648776
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
temp[temp[2]==max(temp[2])].iloc[0][0]==> 5.0
##################################################
```



Conclusion

In [131]:

```python
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Depth","Sample", "AUC"]

for each in summary:
    x.add_row(each)
```

```
    x.add_row(each)

print(x)
```

```
+-----------------+-------+--------+--------------------+
|    Vectorizer   | Depth | Sample |        AUC         |
+-----------------+-------+--------+--------------------+
|      Bow_RF     |   20  |   5    | 0.5795414112816293 |
|     Tfidf_RF    |   10  |   5    | 0.6056405404493665 |
|      W2v_RF     |   10  |   5    | 0.5389685949807443 |
|   Tfidf_w2v_RF  |   10  |   5    | 0.5563133638441838 |
|     BoW_GBDT    |   5   |   5    | 0.6365878757228611 |
|    Tfidf_GBDT   |   20  |   6    | 0.6041408122430133 |
|     W2v_GBDT    |   5   |   9    | 0.6511915389520438 |
|  Tfidf_w2v_GBDT |   5   |   10   | 0.6472732469931177 |
+-----------------+-------+--------+--------------------+
```

Summary: XGBoost has significantly improved the performance over Random Forest model