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	De
project_id	A unique identifier for the proposed project. **Example:**
project_title	Title of the project. **E** • Art Will Make You • First Gr
project_grade_category	Grade level of students for which the project is targeted. One of the enumerate • Grades • Gra • Gra • Grad
project_subject_categories	One or more (comma-separated) subject categories for the projec following enumerated list • Applied L • Care & • Health & • History & • Literacy & L • Math & • Music & T • Specia • **E** • Music & T • Literacy & Language, Math &
school_state	State where school is located (Two-letter U.S. p (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Posta **Exar**
project_subject_subcategories	One or more (comma-separated) subject subcategories for th **E** • L • Literature & Writing, Social S
project_resource_summary	An explanation of the resources needed for the project. **E** • My students need hands on literacy materials to sensory
project_essay_1	First applicat
project_essay_2	Second applicat
project_essay_3	Third applicat
project_essay_4	Fourth applicat
project_submitted_datetime	Datetime when project application was submitted. **Example:** 201 12:43
teacher_id	A unique identifier for the teacher of the proposed project. bdf8baa8fedef6bfeec7ae4ff

Feature	De
`teacher_prefix`	Teacher's title. One of the following enumerate • • • • • • T
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the sam **Exa**

* 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

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

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

```
C:\Users\RASHU TYAGI\Anaconda3\lib\site-packages\smart_open\ssh.py:34: Use
rWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disabled.
`pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disa
bled.  `pip install paramiko` to suppress')
```

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv(r'D:\Rashu Studies\AppliedAICourse\Assignments\Mandatory Ass
ignments\Mandatory Assignment 3 Donors Choose KNN\train_data.csv')
resource_data = pd.read_csv(r'D:\Rashu Studies\AppliedAICourse\Assignments\Mandatory As
signments\Mandatory Assignment 3 Donors Choose KNN\resources.csv')
```

In [3]:

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

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

In [4]:

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


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40840
39
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


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


project_data.head(2)
```

Out[4]:

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

In [5]:

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

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

Out[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

NOW THE MOST IMPORTANT THING HERE IS THAT YOU SHOULD SPLIT OUR DATA INTO TRAIN AND TEST BEFORE APPLYING ANY FIT TECHNIQUE LIKE BOW OR TFIDF BECAUSE OTHERWISE THERE WILL BE DATA LEAKAGE PROBLEM.ALSO FOR PREPROCESSING LIKE STANDARDIZATION AND NORMALIZATION ALSO WE SHOULD KEEP IN MIND THAT TRAIN TEST SPLIT SHOULD BE DONE BEFORE APPLYING THOSE PREPROCESSING TECHNIQUES

In [6]:

```
# REFER THIS SOUNDCLOUD LINK : https://soundcloud.com/applied-ai-course/leakage-bow-and
-tfidf
```

Train_Test Split

In [7]:

```
# train test split
# note that here This stratify parameter makes a split so that the proportion of values
in the sample produced will be the same as the proportion of values provided to paramet
er stratify.
#For example, if variable y is a binary categorical variable with values 0 and 1 and th
ere are 25% of zeros and 75% of ones, stratify=y will make sure that your random split
 has 25% of 0's and 75% of 1's.

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=0.33, stratify = project_data['project_is_approved'])

# now getting the crossvalidation data from our train data
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, strat
ify=y_train)
```

In [8]:

```
# Now we will be removing the column "project_is_approved" because that is the only one
which our model needs to predict

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

Now we will do all kind of preprocessing required for the train data ,test data,crossvalidation data separately

FOR TRAIN DATA

Preprocessing of `project_subject_categories'

In [9]:

```
catogories = list(X_train['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

from collections import Counter
my_counter = Counter()
for word in X_train['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 [10]:

```
sub_catogories = list(X_train['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

X_train['clean_subcategories'] = sub_cat_list
X_train.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 X_train['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]))
```

Text preprocessing

In [11]:

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

In [12]:

```
X_train.head(2)
```

Out[12]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
19991	1725	p098686	64f27bf380a577cf5653bc4c3a44f350	Teacher	ID
104275	92128	p089992	0af24579feb41d991fe38bb22d9ca360	Ms.	NH

In [13]:

```
# printing some random reviews
print(X_train['essay'].values[0])
print("="*50)
print(X_train['essay'].values[150])
print("="*50)
print(X_train['essay'].values[1000])
print("="*50)
print(X_train['essay'].values[20000])
print("="*50)
print(X_train['essay'].values[9999])
print("="*50)
```

There are 40 students in our fifth grade that are from rural Idaho. Sever
al are ESL (English as a Second Language) students. One third of our chil
dren come from a low SES (socio-economic status) background. Some students
have Individual Education Plans while others need to be challenged. \r\n
\r\n   We are often in need of funding for several projects, supplies an
d curriculum due to minimal funds. Our students are especially creative a
nd love to work on projects that fuel their intellectual curiosity.These c
reative learning materials will give our students a more hands on approach
to learning.   Student engagement is critical in elementary school. The us
e of these materials will provide our students with interactive learnin
g.\r\n\r\n   Reading, math, science, social studies and English languag
e arts will be enhanced with materials used to create things such as DNA s
trands, solar systems, United States interactive notebooks, story boards,
math flip books and more.\r\n   \r\n   With a spending freeze in place
there aren't funds available to purchase the supplies needed for these han
ds on projects that make learning fun and exciting. We appreciate any hel
p given.\r\n\r\nnannan
==
My students are second graders who are very fortunate to be in a technolog
y rich school. As a class we are lucky enough to have a chromebook for eac
h student. These chromebooks allow us to access the latest programs in web
based learning. We also use our chromebooks for practicing skills learned
in the classroom, and for research projects. My children also love to use
the floor for group learning projects. The problem we have is storing and
charging our chromebooks without giving up precious classroom space.A nea
t, organized classroom is necessary for optimal learning. Classroom space
is needed for students, and right now, our classroom space is cluttered wi
th bulky carts and tangled cords to store and charge our chromebooks.\r\n
These specialized charge stations take up very little space, and keep the
cords tangle free. My students will waste no time trying to find their cha
rgers, untangle the cords, and figure out where to put their chromebooks.
Quick transitions will allow more time for using the chromebooks for their
intended purpose of technology enriched learning. There will also be more
classroom space for child centered, small group instruction. \r\n   \r\n
nannan
==
My students come form very diverse background. They are very energetic,
full of wonder, and ready to learn. My goal is to make sure all my student
s are offered the best education possible. I am searching for new ways to
bring technology to my students. I want to provide them with the opportuni
ties to thrive in a culture of technology. I would also like to help them
be proud of the work they do and create a portfolio of the work they are p
roud of.Technology is everywhere! As I continue to learn like my students,
I am learning about different interactive ways to make learning fun. With
your help I would like to have an independent Ipad center, where my studen
ts can create their own portfolio of the work they are proud of with an ap
p called SeeSaw\r\n\r\nMy students will benefit from an IPad in the classr
oom because they will be able to practice what they learn at their own pac
e without embarrassment. Using SeeSaw, they will also be able to create a
portfolio of the work they are proud of, which in turn will make them more
accountable and excited to work and learn.\r\n\r\n \r\n\r\n\r\n\r\nnannan
==
My third grade students love technology. They want to use it each and ever
yday in our classroom. Some students do and do not have access to technolo
gy when not in school. This is a disadvantage for them because our word is
constantly growing in technology and they need to be up to par for succes
s.\r\n\r\nMany are from low income families. In addition, majority of them
are ELL learners with English being their second language. They enjoy read
ing many different types of materials (books, magazines, newspapers, onlin
e articles). They are also active in technology. Each student has access t
o Chromebook in our 3rd grade community, however at all times they may or

```
may not work. Within our school, all of the students receive a free breakf
ast each and everyday. They also have plenty of after school activities fo
r students to join and be apart of. There is even a club for parents to he
lp them get jobs, learn English.I can envision all of my students being ab
le to learn and transform using OS system functions. Students have access
to Windows and Chrome operating system, but what about OS/Apple? Students
need to be exposed to all operating systems. Who knows when in their life
when they will have to use an OS system, so why not expose them to it now.
Students will be able to access all of their accounts on this Macbook such
as Myon, Spelling-City, I-Ready, Study Island, and even Apple made applica
tions such as I-Movie to get more creative in reading class (recording pla
ys, editing them together, etc), are just to name a few ideas!\r\n\r\nThis
project will make a difference because now students will have access to AL
L modes of technology in our classroom.\r\nStudents will be able to access
and explore various Apple application on the Macbook and even create magic
al, educational items! This materials can benefit my students in the long
run because it could open many doors that may be closed for them!nannan
==================================================
Our school is a Title I school that serves approximately 700 students in P
re-K-fifth grade.  This past school year we were named a Texas honor roll
campus for consistent student performance and achievement.  We use the Lea
der in Me program in addition to our curriculum to help our students acqui
re real life, as well as academic skills and promote student responsibilit
y and accountability.  Students are given numerous opportunities to synerg
ize with their peers, problem solve and use their new found knowledge to c
reate projects and share within the campus and community.Students in the f
irst grade at my school have a grade level goal to reach 30 accelerated re
ader points by the end of the year to receive a reading trophy.  Students
are working very hard daily to read books and reach their goal.  The probl
em we currently have is finding enough books on their level.  Many of my s
tudents are beginning readers and need lower level books.  \r\nMy students
are so excited and motivated to reach their reading goal.  I want to make
it more attainable by providing as many books on their level as possible.
I can already see fluency and confidence blooming with the little we have
done.nannan
==================================================
```

In [14]:

```
# creating a function named as decontracted which does the job of decontraction

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

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

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

In [15]:

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

```
My third grade students love technology. They want to use it each and ever
yday in our classroom. Some students do and do not have access to technolo
gy when not in school. This is a disadvantage for them because our word is
constantly growing in technology and they need to be up to par for succes
s.\r\n\r\nMany are from low income families. In addition, majority of them
are ELL learners with English being their second language. They enjoy read
ing many different types of materials (books, magazines, newspapers, onlin
e articles). They are also active in technology. Each student has access t
o Chromebook in our 3rd grade community, however at all times they may or
may not work. Within our school, all of the students receive a free breakf
ast each and everyday. They also have plenty of after school activities fo
r students to join and be apart of. There is even a club for parents to he
lp them get jobs, learn English.I can envision all of my students being ab
le to learn and transform using OS system functions. Students have access
to Windows and Chrome operating system, but what about OS/Apple? Students
need to be exposed to all operating systems. Who knows when in their life
when they will have to use an OS system, so why not expose them to it now.
Students will be able to access all of their accounts on this Macbook such
as Myon, Spelling-City, I-Ready, Study Island, and even Apple made applica
tions such as I-Movie to get more creative in reading class (recording pla
ys, editing them together, etc), are just to name a few ideas!\r\n\r\nThis
project will make a difference because now students will have access to AL
L modes of technology in our classroom.\r\nStudents will be able to access
and explore various Apple application on the Macbook and even create magic
al, educational items! This materials can benefit my students in the long
run because it could open many doors that may be closed for them!nannan
==================================================
```

In [16]:

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

```
My third grade students love technology. They want to use it each and ever
yday in our classroom. Some students do and do not have access to technolo
gy when not in school. This is a disadvantage for them because our word is
constantly growing in technology and they need to be up to par for succes
s.    Many are from low income families. In addition, majority of them are
ELL learners with English being their second language. They enjoy reading
many different types of materials (books, magazines, newspapers, online ar
ticles). They are also active in technology. Each student has access to Ch
romebook in our 3rd grade community, however at all times they may or may
not work. Within our school, all of the students receive a free breakfast
each and everyday. They also have plenty of after school activities for st
udents to join and be apart of. There is even a club for parents to help t
hem get jobs, learn English.I can envision all of my students being able t
o learn and transform using OS system functions. Students have access to W
indows and Chrome operating system, but what about OS/Apple? Students need
to be exposed to all operating systems. Who knows when in their life when
they will have to use an OS system, so why not expose them to it now. Stud
ents will be able to access all of their accounts on this Macbook such as
Myon, Spelling-City, I-Ready, Study Island, and even Apple made applicatio
ns such as I-Movie to get more creative in reading class (recording plays,
editing them together, etc), are just to name a few ideas!    This project
will make a difference because now students will have access to ALL modes
of technology in our classroom.  Students will be able to access and explo
re various Apple application on the Macbook and even create magical, educa
tional items! This materials can benefit my students in the long run becau
se it could open many doors that may be closed for them!nannan
```

In [17]:

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

```
My third grade students love technology They want to use it each and every
day in our classroom Some students do and do not have access to technology
when not in school This is a disadvantage for them because our word is con
stantly growing in technology and they need to be up to par for success Ma
ny are from low income families In addition majority of them are ELL learn
ers with English being their second language They enjoy reading many diffe
rent types of materials books magazines newspapers online articles They ar
e also active in technology Each student has access to Chromebook in our 3
rd grade community however at all times they may or may not work Within ou
r school all of the students receive a free breakfast each and everyday Th
ey also have plenty of after school activities for students to join and be
apart of There is even a club for parents to help them get jobs learn Engl
ish I can envision all of my students being able to learn and transform us
ing OS system functions Students have access to Windows and Chrome operati
ng system but what about OS Apple Students need to be exposed to all opera
ting systems Who knows when in their life when they will have to use an OS
system so why not expose them to it now Students will be able to access al
l of their accounts on this Macbook such as Myon Spelling City I Ready Stu
dy Island and even Apple made applications such as I Movie to get more cre
ative in reading class recording plays editing them together etc are just
to name a few ideas This project will make a difference because now studen
ts will have access to ALL modes of technology in our classroom Students w
ill be able to access and explore various Apple application on the Macbook
and even create magical educational items This materials can benefit my st
udents in the long run because it could open many doors that may be closed
for them nannan
```

In [18]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# because although they are in this list but they matter a lot because
# they change the meaning of the entire sentence.
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', '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', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [19]:

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

```
100%|██████████████████████████████████████████████████████████████████████████████| 49041/49041 [00:22<00:00, 2218.67it/s]
```

In [20]:

```
# after preprocesing of project essays
preprocessed_essays_Train[20000]
```

Out[20]:

'third grade students love technology want use everyday classroom students not access technology not school disadvantage word constantly growing technology need par success many low income families addition majority ell learners english second language enjoy reading many different types materials books magazines newspapers online articles also active technology student access chromebook 3rd grade community however times may may not work within school students receive free breakfast everyday also plenty school activities students join apart even club parents help get jobs learn english envision students able learn transform using os system functions students access windows chrome operating system os apple students need exposed operating systems knows life use os system not expose students able access accounts macbook myon spelling city ready study island even apple made applications movie get creative reading class recording plays editing together etc name ideas project make difference students access modes technology classroom students able access explore various apple application macbook even create magical educational items materials benefit students long run could open many doors may closed nannan'

Preprocessing of project_title

Now we will simply apply the above preprocessing steps on the project title for the train data as well,as it is also a text feature

In [21]:

```
# printing some random titles.
print(X_train['project_title'].values[0])
print("="*50)
print(X_train['project_title'].values[150])
print("="*50)
print(X_train['project_title'].values[1000])
print("="*50)
print(X_train['project_title'].values[20000])
print("="*50)
print(X_train['project_title'].values[9999])
print("="*50)
```

```
Creative Project Cupboard
==================================================
A Charging Station Gives A Charge To Our Chromebooks
==================================================
Technology is better than worksheets!
==================================================
An Apple a Day...Laptop!
==================================================
Growing Readers to Become Leaders
==================================================
```

We have already written the preprocessing codes for different preprocessing approaches now we will simply use those codes on the project titles

In [22]:

```
preprocessed_project_titles_Train = []

for t in tqdm(X_train["project_title"]):
    title = decontracted(t)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_project_titles_Train.append(title.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████████
████| 49041/49041 [00:01<00:00, 47878.15it/s]
```

In [23]:

```
# printing some random titles of train dataset after preprocessing

print(preprocessed_project_titles_Train[5000])
print("="*50)
print(preprocessed_project_titles_Train[7000])
print("="*50)
print(preprocessed_project_titles_Train[10000])
print("="*50)
print(preprocessed_project_titles_Train[45000])
print("="*50)
print(preprocessed_project_titles_Train[22000])
print("="*50)
```

```
scrutinizing scientists
==================================================
heads up
==================================================
ipads first grade
==================================================
reading giving children wings
==================================================
invaluable ipad
==================================================
```

Test Data

Preprocessing of project_subject_categories

In [24]:

```
catogories = list(X_test['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

from collections import Counter
my_counter = Counter()
for word in X_test['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 [25]:

```
sub_catogories = list(X_test['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

X_test['clean_subcategories'] = sub_cat_list
X_test.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 X_test['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]))
```

Text Preprocessing

In [26]:

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

In [27]:

```
X_test.head(5)
```

Out[27]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
80901	52945	p215073	dbb3969f011e07bb7680dd2ef03e8ec2	Ms.	OH
11577	123088	p012077	be17b6b05ba0ce6907eadd83c8e2d066	Ms.	MS
2181	19822	p056209	1e97230203b7e4e5d985533059b4840f	Mrs.	RI
98232	129776	p076610	e2629187abbb81057944e9c51254bb59	Ms.	NY
51939	9335	p050102	3de4b606764d3a6bdb26b304f8c26525	Mrs.	NY

In [28]:

```
# printing some random reviews
print(X_test['essay'].values[0])
print("="*50)
print(X_test['essay'].values[150])
print("="*50)
print(X_test['essay'].values[1000])
print("="*50)
print(X_test['essay'].values[20000])
print("="*50)
print(X_test['essay'].values[9999])
print("="*50)
```

My students are Special needs students who have multiple disabilities. The
y walk into our classroom every day full of life, ready to learn, and exci
ted for what is in store for them that day. They are a group of children w
ho work hard at being an inspiration to themselves and to others. \r\n\r
\nThey enjoy incorporating what they have learned into real life experienc
es and they are very enthusiastic about learning! There are many outside f
actors that they cannot control that can affect their day. They each have
their own learning needs which makes it very important for me to provide m
y students with differentiated instruction, a safe environment as well as
an education that is based around whole brain learning. \r\n\r\nWe are a p
art of an elementary school in Columbus, Ohio which is highest poverty. Ou
r students do not have the resources at home to succeed in this world, so
it is my job to help them learn these skills at school!My students need ha
nds on materials to bring STEAM into our classroom! My students will learn
to think like real scientists & engineers. These kits are specially develo
ped to introduce kids to the STEM design process as a way to solve problem
s—students will plan, build, test and then improve their designs until the
y are successful.\r\nThe Fairy Tales STEAM Kit combines classic stories wi
th hands-on STEM and literacy activities to get students involved! As chil
dren read the stories, animate the puppets, answer questions and complete
each STEM challenge, they'll build critical language & literacy skills, bo
ost creative expression, master early engineering concepts and much more!n
annan
==========
My students are eager to learn how to read. 100 percent of the students at
my school receive free breakfast and lunch throughout the school year as w
ell as during the summer months. My students come from low-economic famili
es and most of them receive a backpack with school supplies at the beginni
ng of the school year.\r\nMy school has been in this neighborhood for over
50 years. Generations have attended this school. They are very proud and l
ove to learn.Your generous donation to our project will improve our Kinder
classroom by building stronger readers. This will change our students' liv
es for the better because they will love school and feel successful at an
early age. The children will be able to choose a book that they may not be
able to read on their own yet and be able to follow along, gaining confid
ence as they go. \r\nDo you remember when your parents read you a book and
you fell in love with that story? We will use our listening center everyd
ay during work stations to stimulate them just as your parents did for yo
u. \r\nMy students will practice reading along with the stories and they
will be able to see their own reading skills improving each day while havi
ng fun and learning!nannan
==========
My First Grade Super Star Learners ARE the leaders of the future. Every Se
ptember I know that First Grade is building the foundation for them to be
future teachers, lawyers, doctors, authors, musicians, and a plethora of o
ther careers that may even be within their hearts as a goal now.\r\n
My students' backgrounds make them an asset to our city; even at 6 years o
ld! They are racially, linguistically, and socioeconomically diverse, and
this should be seen as inherent knowledge that will support and guide thei
r academic learning.The Classroom Carpet is going to be essential to the l
earning of my First Graders. They deserve a comfortable large rug that wil
l allow them both personal space, and a place to participate in whole grou
p learning. \r\n   Especially at this age, (6- 7 year olds), need lea
rning materials that are going to help build motivation to meet, and even
surpass the learning goals set forth in and beyond the classroom. This wi
ll be possible as there are many items, such as the Laptop and Lock and Ke
y Phonics Games, that lend to their literacy and math development in a fu
n and engaging way!\r\n   In addition, their published stories will be
able to come to life with the help of the laptop, and the wireless color p
rinter. The Communication Center will also allow me to quickly inform thei
r families about daily skills taught in the classroom, and ways that they

can supplement their students' learning at home.nannan
==========
My 1st grade class is very excited to be in my classroom! They come into my classroom excited to learn daily! We have been together in 1st grade for 7 months!! I am very excited to see the progress that this class has made in Reading and Math!! They are very excited to be able to read books and work on reading games!\r\n\r\nMy students are mostly from lower income families where their parents work one or two jobs to support their families. These students appreciate the littlest things that make them happy. I want to have a learning environment that will encourage success in school! By having a nice storage unit like this one from Childcraft, my students will be better prepared to learn by having a nice organize work space in our classroom!! My students are constantly losing crayons or pencils because of having no organized work space! The students will know exactly were to go to get their crayons or pencils and put all finished work!!\r\n\r\nAs a 1st grade teacher, I want to show my students how important it is to be able to know exactly were their school supplies are and where to put finished work..this is a great way to teach responsibility to my students in my classroom!nannan
==========
Emotional needs run high with the student population I serve! I am a Special Education Counselor and work with mainly High School and Middle School students. These years are full of highs and lows and can greatly effect a person's self-esteem. The students who qualify for my services can be students who have been through a trauma, living with an impairment, or just suffering from a life situation of any type. Whatever the need is, it is always my goal to help them through it and discover the wonderful person they are regardless of what they have been through or what has been done to them. Art is a vehicle to self-awareness and can be a powerful escape to a positive place. \r\n   The stories they share are heartbreaking and that makes them even more courageous to me when they plow through with success! I mainly work with middle school and high school but can be called to visit even the smallest of students we serve at the pre-k level.Art therapy is a technique used during counseling sessions that works really well for a traveling counselor. I have what some students refer to as my \"Mary Poppins\" bag. I carry around art supplies and an iPad to help my students get relaxed, express their feelings and feel comfortable enough to open up when they are stressed. I feel like my students need art because at the middle school and high school level sometimes saying what you need to say is harder to figure out. This quote from Georgia O'Keeffe expresses the feeling exactly.\r\n\r\n\"I found I could say things with color and shapes that I couldn't say any other way - things I had no words for.\" Georgia O'Keeffe \r\n\r\nSelf-expression is critical in counseling. Expression through art can help a student self-reflect and grow with new understandings of themselves and how they fit into the world around them. \r\nMy goal is to help every single student I meet with and art gives the choice to the student to go wherever they want to go. It's a beautiful thing to watch someone create something they are proud of and I consider it a great joy to be able to be a part of that process.\r\nThese art supplies are the choices of the students I see and they will be greatly appreciated and well used.\r\n\r\n\r\n\r\n\r\nnannan
==========

In [29]:

```
# creating a function named as decontracted which does the job of decontraction

# 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 [30]:

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

```
My 1st grade class is very excited to be in my classroom!  They come into
my classroom excited to learn daily!  We have been together in 1st grade f
or 7 months!!  I am very excited to see the progress that this class has m
ade in Reading and Math!!  They are very excited to be able to read books
and work on reading games!\r\n\r\nMy students are mostly from lower income
families where their parents work one or two jobs to support their familie
s.  These students appreciate the littlest things that make them happy. I
want to have a learning environment that will encourage success in school!
By having a nice storage unit like this one from Childcraft, my students
will be better prepared to learn by having a nice organize work space in o
ur classroom!! My students are constantly losing crayons or pencils becaus
e of having no organized work space! The students will know exactly were t
o go to get their crayons or pencils and put all finished work!!\r\n\r\nAs
a 1st grade teacher, I want to show my students how important it is to be
able to know exactly were their school supplies are and where to put finis
hed work..this is a great way to teach responsibility to my students in my
classroom!nannan
==================================================
```

In [31]:

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

```
My 1st grade class is very excited to be in my classroom!  They come into
my classroom excited to learn daily!  We have been together in 1st grade f
or 7 months!!  I am very excited to see the progress that this class has m
ade in Reading and Math!!  They are very excited to be able to read books
and work on reading games!    My students are mostly from lower income fam
ilies where their parents work one or two jobs to support their families.
These students appreciate the littlest things that make them happy. I want
to have a learning environment that will encourage success in school!By ha
ving a nice storage unit like this one from Childcraft, my students  will
be better prepared to learn by having a nice organize work space in our cl
assroom!! My students are constantly losing crayons or pencils because of
having no organized work space! The students will know exactly were to go
to get their crayons or pencils and put all finished work!!    As a 1st gr
ade teacher, I want to show my students how important it is to be able to
know exactly were their school supplies are and where to put finished wor
k..this is a great way to teach responsibility to my students in my classr
oom!nannan
```

In [32]:

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

```
My 1st grade class is very excited to be in my classroom They come into my
classroom excited to learn daily We have been together in 1st grade for 7
months I am very excited to see the progress that this class has made in R
eading and Math They are very excited to be able to read books and work on
reading games My students are mostly from lower income families where thei
r parents work one or two jobs to support their families These students ap
preciate the littlest things that make them happy I want to have a learnin
g environment that will encourage success in school By having a nice stora
ge unit like this one from Childcraft my students will be better prepared
to learn by having a nice organize work space in our classroom My students
are constantly losing crayons or pencils because of having no organized wo
rk space The students will know exactly were to go to get their crayons or
pencils and put all finished work As a 1st grade teacher I want to show my
students how important it is to be able to know exactly were their school
supplies are and where to put finished work this is a great way to teach r
esponsibility to my students in my classroom nannan
```

In [33]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# because although they are in this list but they matter a lot because
# they change the meaning of the entire sentence.
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', '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', "doesn'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 [34]:

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

```
100%|██████████████████████████████████████████████████████████████████████████████
██████| 36052/36052 [00:16<00:00, 2200.93it/s]
```

In [35]:

```
# after preprocesing of project essays
preprocessed_essays_Test[20000]
```

Out[35]:

```
'1st grade class excited classroom come classroom excited learn daily toge
ther 1st grade 7 months excited see progress class made reading math excit
ed able read books work reading games students mostly lower income familie
s parents work one two jobs support families students appreciate littlest
things make happy want learning environment encourage success school nice
storage unit like one childcraft students better prepared learn nice organ
ize work space classroom students constantly losing crayons pencils no org
anized work space students know exactly go get crayons pencils put finishe
d work 1st grade teacher want show students important able know exactly sc
hool supplies put finished work great way teach responsibility students cl
assroom nannan'
```

Preprocessing of project_title

Now we will simply apply the above preprocessing steps on the project title for the test data as well,as it is also a text feature

In [36]:

```
# printing some random titles.
print(X_test['project_title'].values[0])
print("="*50)
print(X_test['project_title'].values[150])
print("="*50)
print(X_test['project_title'].values[1000])
print("="*50)
print(X_test['project_title'].values[20000])
print("="*50)
print(X_test['project_title'].values[9999])
print("="*50)
```

```
Lets Release Some STEAM!
==================================================
\"Help us to become independent and confident readers.\"
==================================================
Support Our City's Next Leaders- It Starts in First Grade!
==================================================
Got Storage??
==================================================
Dreamy Art Therapy
==================================================
```

We have already written the preprocessing codes for different preprocessing approaches now we will simply use those codes on the project titles

In [37]:

```
preprocessed_project_titles_Test = []

for t in tqdm(X_test["project_title"]):
    title = decontracted(t)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_project_titles_Test.append(title.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████████| 36052/36052 [00:00<00:00, 47878.70it/s]
```

In [38]:

```
# printing some random titles of train dataset after preprocessing

print(preprocessed_project_titles_Test[5000])
print("="*50)
print(preprocessed_project_titles_Test[7000])
print("="*50)
print(preprocessed_project_titles_Test[10000])
print("="*50)
print(preprocessed_project_titles_Test[4500])
print("="*50)
print(preprocessed_project_titles_Test[22000])
print("="*50)
```

```
using art advocate humanitarian rights science
==================================================
global minds need global resources
==================================================
spin art makes you smart
==================================================
a kindergarten stem grow on
==================================================
creativity critical thinking through technology
==================================================
```

Cross validation data

Preprocessing of project_subject_categories

In [39]:

```
catogories = list(X_cv['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

from collections import Counter
my_counter = Counter()
for word in X_test['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 [40]:

```
sub_catogories = list(X_cv['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & 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:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

X_cv['clean_subcategories'] = sub_cat_list
X_cv.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 X_cv['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]))
```

Text Preprocessing

In [41]:

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

In [42]:

```
X_cv.head(5)
```

Out[42]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
90451	101128	p213866	c1e8aaa13d65bd2a7893658e41c19ce6	Mrs.	NC
7737	160973	p242056	3b10caf0ad6113ac0384e5475144e784	Ms.	WA
99441	39896	p169773	d976261c7d1c2f56cce47db29ab7054b	Mrs.	FL
58678	103422	p022983	dea8f01cc09b5197e3ba816714fe2694	Mrs.	MN
74283	54	p009578	d626d8b2ff909ce856e47395c7cb3837	Mrs.	NC

In [43]:

```
# printing some random reviews
print(X_cv['essay'].values[0])
print("="*50)
print(X_cv['essay'].values[150])
print("="*50)
print(X_cv['essay'].values[1000])
print("="*50)
print(X_cv['essay'].values[20000])
print("="*50)
print(X_cv['essay'].values[9999])
print("="*50)
```

My students are some of the brightest, most inspirational children I've ev
er had the pleasure of working with. They come from a variety of backgrou
nds, though all have something happening in their life that marks them as
potentially \"at risk\" for failure in their future. We come from a smal
l, predominantly Hispanic community, with the majority of our school popul
ation falling at or below the poverty line. \r\n\r\nThe more positive edu
cational experiences we can give students in their early years, the highes
t their chances are at succeeding as young adults and into adulthood. With
language barriers, poverty, and cultural differences standing between our
students and many of their educational opportunities, I feel it's vital th
at they be given as much exposure and positive experiences as early as pos
sible.   We must use every opportunity to foster these young minds and en
courage growth and development, as these children are our future.Cultivati
ng learning through sensory play is an intregal part of cognitive and phys
ical development in children younger than 5. Many of our students never w
ill have a chance to experience the beach, or how water moves down a hill,
or how we can build and measure with wet sand versus dry sand. We've used
standard water tables within the classroom for years, but when we began re
modeling our outdoor environment into a naturalized playground, we knew we
needed some way to integrate outside water and sensory play into our daily
routine with a water table that complimented our natural environment as we
ll as build on the play with the addition of gravity as a learning compone
nt.\r\n\r\nYoung children thrive on hands on experiences and through water
and sensory play we can learn reasoning, cause and effect, problem solvin
g, measuring, comparing, and sorting and classifying, just to name a few.
The science possibilities are just as endless, such as melting/freezing an
d concepts such as floating versus sinking. The possibilities are endless
with sensory play. \r\n\r\nnannan
==
I currently have 24 six- and seven-year-old students in my classroom. I am
teaching first graders in a Title I school, located in a high poverty and
high crime area of Pennsylvania.\r\n\r\nAll of my students receive free br
eakfast and lunch. Many of my students come from homes where English is no
t spoken. I have three special education students and two ESL (English as
a Second Language) students in my classroom. Several of the students in my
classroom have an in-class social worker.My students do not always tell th
eir families what we are doing in school. \r\nWe want to involve our famil
ies some more and let them know what we are studying in school.\r\nI would
love to get my first graders lots of writing paper in order for them to wr
ite letters to their families. Not only will this help my students improv
e their writing skills, it will also help their families know what we are
studying.\r\nI am also hoping to get my students Math, Writing and Social
Studies workbooks. These books will help my students review and learn mor
e about these subjects. They will also write to their families about what
they are learning in these books.\r\nThanks so much for your support.\r\nn
annan
==
My students are intelligent, funny, motivated, and curious. They have dre
ams of a future that include college, and are the reason I wake up in the
morning excited to get to work. They are budding community leaders, doctor
s, scientists, entrepreneurs, teachers... They are our future! \r\n\r\nI
teach in a community where approximately 30% of households are low-income,
50% are minority, and 14% are Limited English Proficiency. \r\n\r\nMany of
them are being raised households where both parents work long, difficult h
ours to make ends meet, and have responsibilities after school well beyond
what most children face. Several receive a free lunch based on their econ
omic status. These things may prevent them from getting ahead early in lif
e and may not provide them with the life experiences that can give them th
at \"leg up.\" From the minute they walk in the door of my classroom I fo
cus on their potential and growth while they are with me.\r\n\r\n\r\n\r\nA
s a literacy teacher and life-long lover of books, it breaks my heart to h

ear students say, \"I hate reading!\" I firmly believe that they just have n't found the right book yet! Research continually shows us that, \"...the number one indicator of student success is vocabulary. The more students read, the larger their vocabulary, and the more texts they can understand and apply.\" College requires an average of 2 hours of reading per night, and building the stamina required to be successful at this begins in child hood.\r\n\r\nIf funded, I will be able to meet the independent-reading nee ds of all forty-five of my students. I will be able to provide a book for every genre, interest, and at any Lexile necessary to ensure that all stud ents leave my classroom saying, \"I love to read.\" Please help make their future bright by promoting literacy and starting my diverse classroom libr ary. These students are so deserving!nannan

==========

Often, students are stuck in the proverbial box. Following rules and direc tions that tell them exactly what they are supposed to do and how to do i t. I want the library to be a place for the students to move outside of th at box, to use their imagination and create whatever “thing” they can thin k.My students vary widely in their interests and skill sets. I work with students in grades 4th-8th grade in a rural setting. In a small town like mine, the school is the heart of the community. Not only is it used for s chool, but it is also used for sporting events, family reunions, meetings, and other extracurricular activities. The students love their school and t ake pride in it. \r\nI plan to add on to the Makerspace environment that I have built in my library. I envision a space where students can come down and create, explore, and experiment in an environment where they are not r estricted by being limited to what they are expected to do. In my Makersp ace environment, students will be given the opportunity to use their imagi nation and create using the wide variety of materials I am requesting. Not only will students be able to use their imagination, but they will also be learning problem-solving skills as well. \r\nThrough my Makerspace environ ment, my students have started to develop independence. They are asking l ess and less \"How do you do this?,\" but rather I hear \"Here. Let me sh ow you how you can use that.\" The Makerspace open-ended projects and cha llenges are opening up a new world for my students. Using the whiteboard or chalkboard paint on the library tables will give my students the opport unity to brainstorm their ideas and easily make changes as the group devel ops their idea.\r\n

==========

My 2nd grade students come from all parts of the world, speak a variety of languages and bring different experiences to our class. They are a very s weet group of kids who love helping each other and me! As a new teacher t o the school I have already had a lot to learn from them. I have been so i mpressed with their giving nature, ability to switch from one language to another in a matter of seconds, and thoughtfulness. \r\n\r\n Like mos t 2nd graders they are full of life, wonder and a desire to be included. They are eager to connect with each other and learn whatever I present to them. Their favorite times of the day are when we are sitting together sh aring stories, thoughts on a book, working in groups to achieve a task, an d of course any chance to wiggle or move about. \r\n Though we've onl y been together a week it is safe to say our little class has quickly beco me a little family. I'm very excited for all the learning we will togethe r do this year.We are in need of a comfortable place to gather, learn from each other and build community. A classroom carpet will give us a space t o be together, see each other and hear each other. Being able to do so wi ll give each student the feeling of belonging and security. The Hokki sto ols will satisfy their need to move. I'm sure you can remember being clas s as kid, or even as an adult being in meetings sitting on hard ridged cha irs focusing more on the clock than the content. These stools will allow m y students to safely wiggle and squirm while they work so they can engage with the content. \r\n This project will provide my students with an o ptimal learning environment that puts their emotional and physical needs f

```
irst.   Our carpet will serve as the anchor to our classroom.  We will gat
her on the carpet every morning for morning meeting and at the beginning o
f each lesson for a quick introduction to our learning.  Students will als
o be able to work here with clipboards during small group and independent
learning time.  During independent reading they will also be able to get c
omfy on our carpet.  The wobble stools will allow them to learn in comfort
ably and safely while not restricting them.  Children will be to use the H
okki Stools during independent learning time and also during small groups.
Four of the stools will be at my teacher table to for use during guided le
arning and the other two will be available to use around the room.nannan
==================================================
```

In [44]:

```
# creating a function named as decontracted which does the job of decontraction

# 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 [45]:

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

```
Often, students are stuck in the proverbial box. Following rules and direc
tions that tell them exactly what they are supposed to do and how to do i
t. I want the library to be a place for the students to move outside of th
at box, to use their imagination and create whatever “thing” they can thin
k.My students vary widely in their interests and skill sets.  I work with
students in grades 4th-8th grade in a rural setting.  In a small town like
mine, the school is the heart of the community.  Not only is it used for s
chool, but it is also used for sporting events, family reunions, meetings,
and other extracurricular activities. The students love their school and t
ake pride in it. \r\nI plan to add on to the Makerspace environment that I
have built in my library.  I envision a space where students can come down
and create, explore, and experiment in an environment where they are not r
estricted by being limited to what they are expected to do.  In my Makersp
ace environment, students will be given the opportunity to use their imagi
nation and create using the wide variety of materials I am requesting. Not
only will students be able to use their imagination, but they will also be
learning problem-solving skills as well. \r\nThrough my Makerspace environ
ment, my students have started to develop independence.  They are asking l
ess and less \"How do you do this?,\" but rather I hear \"Here.  Let me sh
ow you how you can use that.\"  The Makerspace open-ended projects and cha
llenges are opening up a new world for my students.  Using the whiteboard
or chalkboard paint on the library tables will give my students the opport
unity to brainstorm their ideas and easily make changes as the group devel
ops their idea.\r\n
==================================================
```

In [46]:

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

```
Often, students are stuck in the proverbial box. Following rules and direc
tions that tell them exactly what they are supposed to do and how to do i
t. I want the library to be a place for the students to move outside of th
at box, to use their imagination and create whatever “thing” they can thin
k.My students vary widely in their interests and skill sets.  I work with
students in grades 4th-8th grade in a rural setting.  In a small town like
mine, the school is the heart of the community.  Not only is it used for s
chool, but it is also used for sporting events, family reunions, meetings,
and other extracurricular activities. The students love their school and t
ake pride in it.   I plan to add on to the Makerspace environment that I h
ave built in my library.  I envision a space where students can come down
and create, explore, and experiment in an environment where they are not r
estricted by being limited to what they are expected to do.  In my Makersp
ace environment, students will be given the opportunity to use their imagi
nation and create using the wide variety of materials I am requesting. Not
only will students be able to use their imagination, but they will also be
learning problem-solving skills as well.   Through my Makerspace environme
nt, my students have started to develop independence.  They are asking les
s and less  How do you do this?,  but rather I hear  Here.  Let me show yo
u how you can use that.   The Makerspace open-ended projects and challenge
s are opening up a new world for my students.  Using the whiteboard or cha
lkboard paint on the library tables will give my students the opportunity
to brainstorm their ideas and easily make changes as the group develops th
eir idea.
```

In [47]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
# because although they are in this list but they matter a lot because
# they change the meaning of the entire sentence.
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', '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', "doesn'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 [48]:

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

```
100%|██████████| 24155/24155 [00:11<00:00, 2158.91it/s]
```

In [49]:

```
# after preprocesing of project essays
preprocessed_essays_Test[20000]
```

Out[49]:

```
'1st grade class excited classroom come classroom excited learn daily toge
ther 1st grade 7 months excited see progress class made reading math excit
ed able read books work reading games students mostly lower income familie
s parents work one two jobs support families students appreciate littlest
things make happy want learning environment encourage success school nice
storage unit like one childcraft students better prepared learn nice organ
ize work space classroom students constantly losing crayons pencils no org
anized work space students know exactly go get crayons pencils put finishe
d work 1st grade teacher want show students important able know exactly sc
hool supplies put finished work great way teach responsibility students cl
assroom nannan'
```

Preprocessing of project_title

Now we will simply apply the above preprocessing steps on the project title for the Cross Validation data as well,as it is also a text feature

In [50]:

```
# printing some random titles.
print(X_cv['project_title'].values[0])
print("="*50)
print(X_cv['project_title'].values[150])
print("="*50)
print(X_cv['project_title'].values[1000])
print("="*50)
print(X_cv['project_title'].values[20000])
print("="*50)
print(X_cv['project_title'].values[9999])
print("="*50)
```

```
Cultivating Sensory Perception Through Water Play
==================================================
Let's Write to Our Families!
==================================================
\"Reading Is Dreaming With Open Eyes.\"
==================================================
\"Make\" a Difference
==================================================
Successful Seating for Squirmy Second Graders
==================================================
```

In [51]:

```
preprocessed_project_titles_Cv = []

for t in tqdm(X_cv["project_title"]):
    title = decontracted(t)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_project_titles_Cv.append(title.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████
████| 24155/24155 [00:00<00:00, 45185.85it/s]
```

In [52]:

```
# printing some random titles of crossvalidation dataset after preprocessing

print(preprocessed_project_titles_Cv[5000])
print("="*50)
print(preprocessed_project_titles_Cv[7000])
print("="*50)
print(preprocessed_project_titles_Cv[10000])
print("="*50)
print(preprocessed_project_titles_Cv[4500])
print("="*50)
print(preprocessed_project_titles_Cv[22000])
print("="*50)
```

```
aspire achieve greater level reading
==================================================
3rd grade authors need writing tools
==================================================
service with smile
==================================================
making math visible equitable access graphing calculators
==================================================
tech bound
==================================================
```

Preparing Data For Models

In [53]:

```
project_data.columns
```

Out[53]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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_approve
d'],
      dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data
- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)
- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

Now firstly we will be vectorizing the categorical data

For vectorizing the categorical data we will be using One Hot Encoding Technique

One Hot Encoding Of Project Clean Categories

Checking if the characters like (. , _ , -) are present in myu dictionary keys or not if they are present I need to reemove them as they don't make any sense for the vocabolary

In [54]:

```
message = "yes . is present " if '.' in sorted_cat_dict else "not . is not present"
print(message)
```

```
not . is not present
```

In [55]:

```
message = "yes - is present " if '-' in sorted_cat_dict else "not - is not present"
print(message)
```

```
not - is not present
```

In [56]:

```
message = "yes _ is present " if '_' in sorted_cat_dict else "not _ is not present"
print(message)
```

```
not _ is not present
```

In [57]:

```
# we use count vectorizer to convert the values into one hot encoded features


from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
binary=True)

# we will be using the X_train for fitting our model because that is the only data a us
er knows rest all are for testing purposes
vectorizer.fit(X_train['clean_categories'].values)

print(vectorizer.get_feature_names())

categories_one_hot_train = vectorizer.transform(X_train['clean_categories'].values)

categories_one_hot_test = vectorizer.transform(X_test['clean_categories'].values)

categories_one_hot_cv = vectorizer.transform(X_cv['clean_categories'].values)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
```

In [58]:

```
print("Shape of  Train data matrix after one hot encoding ",categories_one_hot_train.sh
ape)
print("Shape of  Test data matrix after one hot encoding ",categories_one_hot_test.shap
e)
print("Shape of  CV data matrix after one hot encoding ",categories_one_hot_cv.shape)
```

```
Shape of  Train data matrix after one hot encoding  (49041, 9)
Shape of  Test data matrix after one hot encoding  (36052, 9)
Shape of  CV data matrix after one hot encoding  (24155, 9)
```

One Hot Encoding Of Cleaned Project Sub Category

Checking if the characters like (. , _ , -) are present in myu dictionary keys or not if they are present I need to reemove them as they don't make any sense for the vocabolary

In [59]:

```
message = "yes . is present " if '.' in sorted_sub_cat_dict else "not . is not present"
print(message)
```

```
not . is not present
```

In [60]:

```
message = "yes - is present " if '-' in sorted_sub_cat_dict else "not - is not present"
print(message)
```

```
not - is not present
```

In [61]:

```
message = "yes _ is present " if '_' in sorted_sub_cat_dict else "not _ is not present"
print(message)
```

```
not _ is not present
```

In [62]:

```
# we use count vectorizer to convert the values into one hot encoded features


from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fal
se, binary=True)

# we will be using the X_train for fitting our model because that is the only data a us
er knows rest all are for testing purposes
vectorizer.fit(X_train['clean_subcategories'].values)

print(vectorizer.get_feature_names())

subcategories_one_hot_train = vectorizer.transform(X_train['clean_subcategories'].value
s)

subcategories_one_hot_test = vectorizer.transform(X_test['clean_subcategories'].values)

subcategories_one_hot_cv = vectorizer.transform(X_cv['clean_subcategories'].values)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'Extracurricular',
'ParentInvolvement', 'Civics_Government', 'ForeignLanguages', 'NutritionEd
ucation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'CharacterEducation',
'TeamSports', 'PerformingArts', 'Other', 'College_CareerPrep', 'History_Ge
ography', 'Music', 'EarlyDevelopment', 'Health_LifeScience', 'ESL', 'Gym_F
itness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'Applied
Sciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literac
y']
```

In [63]:

```
print("Shape of  Train data matrix after one hot encoding ",subcategories_one_hot_train
.shape)
print("Shape of  Test data matrix after one hot encoding ",subcategories_one_hot_test.s
hape)
print("Shape of  CV data matrix after one hot encoding ",subcategories_one_hot_cv.shape
)
```

```
Shape of  Train data matrix after one hot encoding  (49041, 30)
Shape of  Test data matrix after one hot encoding  (36052, 30)
Shape of  CV data matrix after one hot encoding  (24155, 30)
```

One hot encoding of teacher prefix

In [64]:

```
mylist_teacher_prefix = list(X_train['teacher_prefix'])
```

In [65]:

```
# We are removing the duplicate values from our list of the teacher prefix

# Source :- https://www.w3schools.com/python/python_howto_remove_duplicates.asp

mylist_teacher_prefix_actual_Train = list(dict.fromkeys(mylist_teacher_prefix))
```

In [66]:

```
# removing the nan from the teacher prefix category as there is no such category of tea
cher which exists

mylist_teacher_prefix_actual_Train = [p for p in mylist_teacher_prefix_actual_Train if
str(p) != 'nan']
mylist_teacher_prefix_actual_Train
```

Out[66]:

```
['Teacher', 'Ms.', 'Mrs.', 'Mr.', 'Dr.']
```

Removing '.' from all the teacher prefixes.

In [67]:

```
# code for this from here --> https://stackoverflow.com/questions/8282553/removing-char
acter-in-list-of-strings

mylist_teacher_prefix_actual_Train = ' '.join(mylist_teacher_prefix_actual_Train).repla
ce('8','').split()

print(mylist_teacher_prefix_actual_Train)
```

```
['Teacher', 'Ms.', 'Mrs.', 'Mr.', 'Dr.']
```

In [68]:

```
# we use count vectorizer to convert the values into one hot encoded features

# now we are working on teacher prefix data

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=mylist_teacher_prefix_actual_Train, lowercase=F
alse, binary=True)

# I was getting an error like "np.nan is an invalid document, expected byte or unicode
 string."
# below is the solution

# https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerr
or-np-nan-is-an-invalid-document


vectorizer.fit(X_train['teacher_prefix'].values.astype('U'))
print(vectorizer.get_feature_names())

teacher_prefix_one_hot_train = vectorizer.transform(X_train['teacher_prefix'].values.as
type('U'))
teacher_prefix_one_hot_test = vectorizer.transform(X_test['teacher_prefix'].values.asty
pe('U'))
teacher_prefix_one_hot_cv = vectorizer.transform(X_cv['teacher_prefix'].values.astype(
'U'))
```

```
['Teacher', 'Ms.', 'Mrs.', 'Mr.', 'Dr.']
```

In [69]:

```
print("Shape of matrix of Train data after one hot encoding ",teacher_prefix_one_hot_tr
ain.shape)
print("Shape of matrix of Test data after one hot encoding ",teacher_prefix_one_hot_tes
t.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",teacher_prefix
_one_hot_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (49041, 5)
Shape of matrix of Test data after one hot encoding  (36052, 5)
Shape of matrix of Cross Validation data after one hot encoding  (24155,
5)
```

One Hot encoding of project grade category

In [70]:

```
mylist_project_grade_category = list(X_train['project_grade_category'])
```

In [71]:

```
# We are removing the duplicate values from our list of the project grade category

# Source :- https://www.w3schools.com/python/python_howto_remove_duplicates.asp

mylist_project_grade_category_actual = list(dict.fromkeys(mylist_project_grade_category
))
```

In [72]:

```
type(mylist_project_grade_category_actual)
print(mylist_project_grade_category_actual[0])

n = len(mylist_project_grade_category_actual)
print(n)

# I already saw by running the code that the word Grades is unnecessarily present in th
e elements of list hence trying to remove that word
# how to remove a word from a sentence --> https://codescracker.com/python/program/pyth
on-program-remove-word-from-sentence.htm
for m in range(0,4,1):
    words = mylist_project_grade_category_actual[m].split()
    mylist_project_grade_category_actual[m] = ''.join([j for j in words if j not in "Gr
ades"])
print(mylist_project_grade_category_actual)
```

```
Grades 3-5
4
['3-5', 'PreK-2', '9-12', '6-8']
```

Replacing '-' with '_' as it is the convention to write

In [73]:

```
# code for this from here --> https://stackoverflow.com/questions/8282553/removing-char
acter-in-list-of-strings

mylist_project_grade_category_actual = ' '.join(mylist_project_grade_category_actual).r
eplace('-','_').split()

print(mylist_project_grade_category_actual)
```

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

In [74]:

```
# we use count vectorizer to convert the values into one hot encoded features

# now we are working on project grade category data

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=mylist_project_grade_category_actual, lowercase
=False, binary=True)


vectorizer.fit(X_train['project_grade_category'].values)
print(vectorizer.get_feature_names())

project_grade_categories_one_hot_train = vectorizer.transform(X_train['project_grade_ca
tegory'].values)
project_grade_categories_one_hot_test = vectorizer.transform(X_test['project_grade_cate
gory'].values)
project_grade_categories_one_hot_cv = vectorizer.transform(X_cv['project_grade_categor
y'].values)
```

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

In [75]:

```
print("Shape of matrix of Train data after one hot encoding ",project_grade_categories_
one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",project_grade_categories_o
ne_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",project_grade_
categories_one_hot_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (49041, 4)
Shape of matrix of Test data after one hot encoding  (36052, 4)
Shape of matrix of Cross Validation data after one hot encoding  (24155,
4)
```

One hot encoding of School States

In [76]:

```
type(list(project_data['school_state']))
```

Out[76]:

```
list
```

In [77]:

```
mylist = list(X_train['school_state'])
```

In [78]:

```
# We are removing the duplicate values from our list of the state codes

# Source :- https://www.w3schools.com/python/python_howto_remove_duplicates.asp

mylist_actual = list(dict.fromkeys(mylist))
```

In [79]:

```
# we use count vectorizer to convert the values into one hot encoded features

# now we are working on school state data

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=mylist_actual, lowercase=False, binary=True)

vectorizer.fit(X_train['school_state'].values)

print(vectorizer.get_feature_names())

school_state_categories_one_hot_train = vectorizer.transform(X_train['school_state'].va
lues)
school_state_categories_one_hot_test = vectorizer.transform(X_test['school_state'].valu
es)
school_state_categories_one_hot_cv = vectorizer.transform(X_cv['school_state'].values)
```

```
['ID', 'NH', 'IN', 'DC', 'AZ', 'GA', 'MD', 'AL', 'IL', 'LA', 'NY', 'CA',
'SC', 'NC', 'FL', 'OR', 'OH', 'UT', 'MA', 'NM', 'TX', 'MN', 'WI', 'NJ', 'W
A', 'MI', 'TN', 'PA', 'MO', 'CT', 'VA', 'NV', 'OK', 'HI', 'KY', 'WV', 'S
D', 'IA', 'MS', 'CO', 'AR', 'ND', 'WY', 'ME', 'KS', 'DE', 'AK', 'MT', 'N
E', 'VT', 'RI']
```

In [80]:

```
print("Shape of matrix of Train data after one hot encoding ",school_state_categories_o
ne_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",school_state_categories_on
e_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",school_state_c
ategories_one_hot_cv.shape)
```

```
Shape of matrix of Train data after one hot encoding  (49041, 51)
Shape of matrix of Test data after one hot encoding  (36052, 51)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 5
1)
```

Now as we have done the vectorizing of categorical data now we will be vectorizing the text data using different techniques

Vectorizing the text data

Technique -1 Bag of words(BOW)

Essays

Essays Train Data

In [81]:

```
# I am not setting the value of min_df here because i read here
# https://stackoverflow.com/questions/27697766/understanding-min-df-and-max-df-in-scikit-countvectorizer
# that min_df helps in better performance but might also give poor clusters hence am trying without it this time



vectorizer_project_essay_bow = CountVectorizer(ngram_range=(1, 2),min_df = 10,max_features = 5000)
vectorizer_project_essay_bow.fit(preprocessed_essays_Train)

essay_bow_train = vectorizer_project_essay_bow.transform(preprocessed_essays_Train)

print("Shape of matrix after bag of words ",essay_bow_train.shape)
```

```
Shape of matrix after bag of words  (49041, 5000)
```

Essay Test Data

In [82]:

```
# I am not setting the value of min_df here because i read here
# https://stackoverflow.com/questions/27697766/understanding-min-df-and-max-df-in-scikit-countvectorizer
# that min_df helps in better performance but might also give poor clusters hence am trying without it this time


# now note that the below two lines are wrong because we have already trained the
# model on the training data now using that model we should get the bow representation
# of test data. After all training from test data only and then cecking for its accuracy will ofcourse give
# good accuracy.

# lines not to be used (I used them but then going through the code realised the mistake)

#vectorizer = CountVectorizer()
#vectorizer.fit(preprocessed_essays_Test)

essay_bow_test = vectorizer_project_essay_bow.transform(preprocessed_essays_Test)

print("Shape of matrix after bag of words ",essay_bow_test.shape)
```

```
Shape of matrix after bag of words  (36052, 5000)
```

Essay Cross Validation data

In [83]:

```
# I am not setting the value of min_df here because i read here
# https://stackoverflow.com/questions/27697766/understanding-min-df-and-max-df-in-scikit-countvectorizer
# that min_df helps in better performance but might also give poor clusters hence am trying without it this time

# similarly below two lines should not be used

#vectorizer = CountVectorizer()
#vectorizer.fit(preprocessed_essays_Cv)

essay_bow_cv = vectorizer_project_essay_bow.transform(preprocessed_essays_Cv)

print("Shape of matrix after bag of words ",essay_bow_cv.shape)
```

```
Shape of matrix after bag of words  (24155, 5000)
```

Project Title

In [84]:

```
# we are imposing no min_df or ngram_range constraint on project title vectorization
vectorizer_project_title_bow = CountVectorizer()
```

Bag of words on Project Title Train data

In [85]:

```
vectorizer_project_title_bow.fit(preprocessed_project_titles_Train)
project_title_bow_train = vectorizer_project_title_bow.transform(preprocessed_project_titles_Train)
print("Shape of matrix after bag of words ",project_title_bow_train.shape)
```

```
Shape of matrix after bag of words  (49041, 11657)
```

Bag of words on Project Title Test data

In [86]:

```
project_title_bow_test = vectorizer_project_title_bow.transform(preprocessed_project_titles_Test)
print("Shape of matrix after bag of words ",project_title_bow_test.shape)
```

```
Shape of matrix after bag of words  (36052, 11657)
```

Bag of words on Project Title Cross Validation data

In [87]:

```
project_title_bow_cv = vectorizer_project_title_bow.transform(preprocessed_project_titl
es_Cv)
print("Shape of matrix after bag of words ",project_title_bow_cv.shape)
```

```
Shape of matrix after bag of words  (24155, 11657)
```

Technique-2 TFIDF

Essay Data

Essay Train Data

In [151]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf = TfidfVectorizer(max_features = 5000,ngram_range = (1,2), min_df = 10
)
vectorizer_tfidf.fit(preprocessed_essays_Train)

text_tfidf_train = vectorizer_tfidf.transform(preprocessed_essays_Train)
print("Shape of matrix after tfidf ",text_tfidf_train.shape)

print(type(text_tfidf_train))

# we are converting a dictionary with word as a key, and the idf as a value

dictionary = dict(zip(vectorizer_tfidf.get_feature_names(), list(vectorizer_tfidf.idf_
)))
tfidf_words = set(dictionary.keys())
```

```
Shape of matrix after tfidf  (49041, 5000)
<class 'scipy.sparse.csr.csr_matrix'>
```

Essay Test data

In [152]:

```
text_tfidf_test = vectorizer_tfidf.transform(preprocessed_essays_Test)
print("Shape of matrix after tfidf ",text_tfidf_test.shape)
```

```
Shape of matrix after tfidf  (36052, 5000)
```

Essay Cross Validation Data

In [153]:

```
text_tfidf_cv = vectorizer_tfidf.transform(preprocessed_essays_Cv)
print("Shape of matrix after tfidf ",text_tfidf_cv.shape)
```

```
Shape of matrix after tfidf  (24155, 5000)
```

Project Title

Project title train data

In [154]:

```
vectorizer = TfidfVectorizer(max_features = 3000,ngram_range = (1,2), min_df = 10)

vectorizer.fit(preprocessed_project_titles_Train)

project_title_tfidf_train = vectorizer.transform(preprocessed_project_titles_Train)

print("Shape of matrix after tfidf ",project_title_tfidf_train.shape)

# we are converting a dictionary with word as a key, and the idf as a value

dictionary = dict(zip(vectorizer.get_feature_names(), list(vectorizer.idf_)))
tfidf_project_title_words = set(dictionary.keys())
```

```
Shape of matrix after tfidf  (49041, 3000)
```

Project title test data

In [155]:

```
project_title_tfidf_test = vectorizer.transform(preprocessed_project_titles_Test)
print("Shape of matrix after tfidf ",project_title_tfidf_test.shape)
```

```
Shape of matrix after tfidf  (36052, 3000)
```

Project title cross validation data

In [156]:

```
title_tfidf_cv = vectorizer.transform(preprocessed_project_titles_Cv)
print("Shape of matrix after tfidf ",title_tfidf_cv.shape)
```

```
Shape of matrix after tfidf  (24155, 3000)
```

Technique-3 Average Word to Vector

Using pretrained w2v model in the file glove

In [94]:

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

Essay Train data

In [95]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_train = []; # the avg-w2v for each essay is stored in this list
for sentence in tqdm(preprocessed_essays_Train): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))
```

```
100%|████████████████████████████████████████████████████████████████
██████| 49041/49041 [00:11<00:00, 4301.05it/s]

49041
300
```

Essay Test data

In [96]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_test = []; # the avg-w2v for each essay is stored in this list
for sentence in tqdm(preprocessed_essays_Test): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
██████| 36052/36052 [00:08<00:00, 4213.78it/s]

36052
300
```

Essay Cross Validation data

In [97]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_cv = []; # the avg-w2v for each essay is stored in this list
for sentence in tqdm(preprocessed_essays_Cv): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)

print(len(avg_w2v_vectors_cv))
print(len(avg_w2v_vectors_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
██████| 24155/24155 [00:05<00:00, 4256.80it/s]

24155
300
```

Project Title train data

In [98]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_project_title_train = []; # the avg-w2v for each essay is stored in thi
s list
for sentence in tqdm(preprocessed_project_titles_Train): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_project_title_train.append(vector)

print(len(avg_w2v_vectors_project_title_train))
print(len(avg_w2v_vectors_project_title_train[0]))
```

```
100%|████████████████████████████████████████████████████████████████████
████| 49041/49041 [00:00<00:00, 79439.56it/s]

49041
300
```

Project Title test data

In [99]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_project_title_test = []; # the avg-w2v for each essay is stored in this
list
for sentence in tqdm(preprocessed_project_titles_Test): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_project_title_test.append(vector)

print(len(avg_w2v_vectors_project_title_test))
print(len(avg_w2v_vectors_project_title_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████
████| 36052/36052 [00:00<00:00, 75856.09it/s]

36052
300
```

Project title Cross Validation data

In [100]:

```
# average Word2Vec
# compute average word2vec for each preprocessed essay.

avg_w2v_vectors_project_title_cv = []; # the avg-w2v for each essay is stored in this l
ist
for sentence in tqdm(preprocessed_project_titles_Cv): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_project_title_cv.append(vector)

print(len(avg_w2v_vectors_project_title_cv))
print(len(avg_w2v_vectors_project_title_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████
████| 24155/24155 [00:00<00:00, 79256.99it/s]

24155
300
```

Technique-4 TFIDF weighted Word To Vector

Essay train data

In [101]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_text_train = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_essays_Train): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_train.append(vector)

print(len(tfidf_w2v_vectors_text_train))
print(len(tfidf_w2v_vectors_text_train[0]))
```

```
100%|████████████████████████████████████████████████████████████████
██████| 49041/49041 [00:57<00:00, 858.13it/s]

49041
300
```

Essay test data

In [102]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_text_test = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_essays_Test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_test.append(vector)

print(len(tfidf_w2v_vectors_text_test))
print(len(tfidf_w2v_vectors_text_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████
██████| 36052/36052 [00:41<00:00, 858.47it/s]


36052
300
```

Essay cross validation

In [103]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_text_cv = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_essays_Cv): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_cv.append(vector)

print(len(tfidf_w2v_vectors_text_cv))
print(len(tfidf_w2v_vectors_text_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████████
██████| 24155/24155 [00:28<00:00, 854.07it/s]

24155
300
```

Project Title Train data

In [104]:

```
vectorizer = TfidfVectorizer(max_features = 2000, min_df = 10)

vectorizer.fit(preprocessed_project_titles_Train)

project_title_tfidf_train = vectorizer.transform(preprocessed_project_titles_Train)

print("Shape of matrix after tfidf ",project_title_tfidf_train.shape)

# we are converting a dictionary with word as a key, and the idf as a value

dictionary = dict(zip(vectorizer.get_feature_names(), list(vectorizer.idf_)))
tfidf_project_title_words = set(dictionary.keys())
```

```
Shape of matrix after tfidf  (49041, 2000)
```

In [105]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_project_title_train = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_project_titles_Train): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_project_title_train.append(vector)

print(len(tfidf_w2v_vectors_project_title_train))
print(len(tfidf_w2v_vectors_project_title_train[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████
████| 49041/49041 [00:01<00:00, 42921.61it/s]

49041
300
```

Project title test data

In [106]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_project_title_test = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_project_titles_Test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_project_title_test.append(vector)

print(len(tfidf_w2v_vectors_project_title_test))
print(len(tfidf_w2v_vectors_project_title_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████
████| 36052/36052 [00:00<00:00, 41908.36it/s]


36052
300
```

project title cross validation data

In [107]:

```
# tfidf Word2Vec
# computing average word2vec for each Project Title is stored in this list

tfidf_w2v_vectors_project_title_cv = []; # the tfidf-w2v for each essay
for sentence in tqdm(preprocessed_project_titles_Cv): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if (word in glove_words) and (word in dictionary):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
tence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
tting 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_project_title_cv.append(vector)

print(len(tfidf_w2v_vectors_project_title_cv))
print(len(tfidf_w2v_vectors_project_title_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████
████| 24155/24155 [00:00<00:00, 43404.30it/s]

24155
300
```

Numerical Features

vectorizing numerical features

Price for projects

In [108]:

```
# now firstly we will try to add the price and the quantity of the items required from
 the resource dataframe

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

price_quantity_data.head(2)
```

Out[108]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

now we need to join the above dataframe with our train,test,cv data that we already have

In [109]:

```
X_train = pd.merge(X_train, price_quantity_data, on='id', how='left')
X_test = pd.merge(X_test, price_quantity_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_quantity_data, on='id', how='left')
```

we will be performing the normalization of the numerical data here

Normalizing the price data

In [110]:

```
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['price'].values.reshape(-1,1))

price_train = normalizer.transform(X_train['price'].values.reshape(-1,1))
price_cv = normalizer.transform(X_cv['price'].values.reshape(-1,1))
price_test = normalizer.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(price_train.shape, y_train.shape)
print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [111]:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

scaler.fit(X_train['price'].values.reshape(-1,1))

price_train = scaler.transform(X_train['price'].values.reshape(-1,1))
price_cv = scaler.transform(X_cv['price'].values.reshape(-1,1))
price_test = scaler.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(price_train.shape, y_train.shape)
print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

Normalizing the quantity data

In [112]:

```
from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

# normalizer.fit(X_train['quantity'].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['quantity'].values.reshape(-1,1))

quantity_train = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
quantity_test = normalizer.transform(X_test['quantity'].values.reshape(-1,1))
quantity_cv = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(quantity_train.shape, y_train.shape)
print(quantity_test.shape, y_test.shape)
print(quantity_cv.shape, y_cv.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(36052, 1) (36052,)
(24155, 1) (24155,)
```

In [113]:

```
scaler = StandardScaler()

scaler.fit(X_train['quantity'].values.reshape(-1,1))

quantity_train = scaler.transform(X_train['quantity'].values.reshape(-1,1))
quantity_test = scaler.transform(X_test['quantity'].values.reshape(-1,1))
quantity_cv = scaler.transform(X_cv['quantity'].values.reshape(-1,1))

print("After vectorizations")

print(quantity_train.shape, y_train.shape)
print(quantity_test.shape, y_test.shape)
print(quantity_cv.shape, y_cv.shape)
```

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

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

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

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

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

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

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

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


After vectorizations
(49041, 1) (49041,)
(36052, 1) (36052,)
(24155, 1) (24155,)
```

Normalizing the number of previously posted projects by a teacher

In [114]:

```
normalizer = Normalizer()

# normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].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['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

prev_projects_train = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_projects_cv = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
prev_projects_test = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print("After converting into vectors form")
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
```

```
After converting into vectors form
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

In [115]:

```
scaler = StandardScaler()

scaler.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1
))

prev_projects_train = scaler.transform(X_train['teacher_number_of_previously_posted_pro
jects'].values.reshape(-1,1))
prev_projects_cv = scaler.transform(X_cv['teacher_number_of_previously_posted_projects'
].values.reshape(-1,1))
prev_projects_test = scaler.transform(X_test['teacher_number_of_previously_posted_proje
cts'].values.reshape(-1,1))

print("After converting into vectors form")
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
```

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

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

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

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

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

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

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

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


After converting into vectors form
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

Some of the features still remaining for extracting are Project_title_word_count,project_essay_word_count,Sentiment_score of each essay so let us extract and add them to our dataset first.

Project Title Word Count

Training data

In [116]:

```
project_title_word_count_train = []
for i in range(0,len(preprocessed_project_titles_Train),1):
    project_title_word_count_train.append(len((preprocessed_project_titles_Train[i]).sp
lit()))
```

In [117]:

```
len(project_title_word_count_train)
```

Out[117]:

```
49041
```

In [118]:

```
project_title_word_count_train[49040]
```

Out[118]:

```
3
```

In [119]:

```
import numpy as np
project_title_word_count_train = np.asarray(project_title_word_count_train).reshape(len
(project_title_word_count_train),1)
project_title_word_count_train.shape
```

Out[119]:

```
(49041, 1)
```

Crossvalidation data

In [120]:

```
project_title_word_count_cv = []
for i in range(0,len(preprocessed_project_titles_Cv),1):
    project_title_word_count_cv.append(len((preprocessed_project_titles_Cv[i]).split
()))
```

In [121]:

```
project_title_word_count_cv[490]
```

Out[121]:

```
5
```

In [122]:

```
import numpy as np
project_title_word_count_cv = np.asarray(project_title_word_count_cv).reshape(len(proje
ct_title_word_count_cv),1)
project_title_word_count_cv.shape
```

Out[122]:

```
(24155, 1)
```

Test data

In [123]:

```
project_title_word_count_test = []
for i in range(0,len(preprocessed_project_titles_Test),1):
    project_title_word_count_test.append(len((preprocessed_project_titles_Test[i]).spli
t()))
```

In [124]:

```
project_title_word_count_test[490]
```

Out[124]:

```
6
```

In [125]:

```
import numpy as np
project_title_word_count_test = np.asarray(project_title_word_count_test).reshape(len(p
roject_title_word_count_test),1)
project_title_word_count_test.shape
```

Out[125]:

```
(36052, 1)
```

Project Essay Word Count

Training data

In [126]:

```
project_essay_word_count_train = []
for i in range(0,len(preprocessed_essays_Train),1):
    project_essay_word_count_train.append(len((preprocessed_essays_Train[i]).split()))
```

In [127]:

```
project_essay_word_count_train[490]
```

Out[127]:

```
154
```

In [128]:

```
import numpy as np
project_essay_word_count_train = np.asarray(project_essay_word_count_train).reshape(len
(project_essay_word_count_train),1)
project_essay_word_count_train.shape
```

Out[128]:

```
(49041, 1)
```

Crossvalidation data

In [129]:

```
project_essay_word_count_cv = []
for i in range(0,len(preprocessed_essays_Cv),1):
    project_essay_word_count_cv.append(len((preprocessed_essays_Cv[i]).split()))
```

In [130]:

```
 project_essay_word_count_cv[490]
```

Out[130]:

```
102
```

In [131]:

```
import numpy as np
project_essay_word_count_cv = np.asarray(project_essay_word_count_cv).reshape(len(proje
ct_essay_word_count_cv),1)
project_essay_word_count_cv.shape
```

Out[131]:

```
(24155, 1)
```

Test Data

In [132]:

```
project_essay_word_count_test = []
for i in range(0,len(preprocessed_essays_Test),1):
    project_essay_word_count_test.append(len((preprocessed_essays_Test[i]).split()))
```

In [133]:

```
 project_essay_word_count_test[490]
```

Out[133]:

```
106
```

In [134]:

```
import numpy as np
project_essay_word_count_test = np.asarray(project_essay_word_count_test).reshape(len(p
roject_essay_word_count_test),1)
project_essay_word_count_test.shape
```

Out[134]:

```
(36052, 1)
```

Sentiment Score of each essay

In [135]:

```
# sample working of sentiment analyser

import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

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

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest
students with the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multi
ple intelligences i use a wide range\
of techniques to help all my students succeed students in my class come from a variety
 of different backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school
is a caring community of successful \
learners which can be seen through collaborative student project based learning in and
 out of the classroom kindergarteners \
in my class love to work with hands on materials and have many different opportunities
 to practice a skill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspec
t of the kindergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love
 to role play in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with
real food i will take their idea \
and create common core cooking lessons where we learn important math and writing concep
ts while cooking delicious healthy \
food for snack time my students will have a grounded appreciation for the work that wen
t into making the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this p
roject would expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make hom
emade applesauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create o
ur own cookbooks to be printed and \
shared with families students will gain math and literature skills as well as a life lo
ng enjoyment for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

In [136]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

In [137]:

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

```
[nltk_data] Downloading package vader_lexicon to C:\Users\RASHU
[nltk_data]     TYAGI\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Out[137]:

```
True
```

In [138]:

```
len(preprocessed_essays_Train)
```

Out[138]:

```
49041
```

Training data

In [139]:

```
sid = SentimentIntensityAnalyzer()
```

In [140]:

```
positive_train = []
negative_train = []
neutral_train = []
compound_train = []
for i in tqdm(preprocessed_essays_Train):
    positive_train.append(sid.polarity_scores(i)['pos'])
    negative_train.append(sid.polarity_scores(i)['neg'])
    neutral_train.append(sid.polarity_scores(i)['neu'])
    compound_train.append(sid.polarity_scores(i)['compound'])
```

```
100%|██████████████████████████████████████████████████████████████████████
██████| 49041/49041 [04:34<00:00, 186.10it/s]
```

Crossvalidation data

In [141]:

```
sid = SentimentIntensityAnalyzer()
```

In [142]:

```
positive_cv = []
negative_cv = []
neutral_cv = []
compound_cv = []
for i in tqdm(preprocessed_essays_Cv):
    positive_cv.append(sid.polarity_scores(i)['pos'])
    negative_cv.append(sid.polarity_scores(i)['neg'])
    neutral_cv.append(sid.polarity_scores(i)['neu'])
    compound_cv.append(sid.polarity_scores(i)['compound'])
```

```
100%|████████████████████████████████████████████████████████████████
██████| 24155/24155 [02:11<00:00, 183.91it/s]
```

Test Data

In [143]:

```
sid = SentimentIntensityAnalyzer()
```

In [144]:

```
positive_test = []
negative_test = []
neutral_test = []
compound_test = []
for i in tqdm(preprocessed_essays_Test):
    positive_test.append(sid.polarity_scores(i)['pos'])
    negative_test.append(sid.polarity_scores(i)['neg'])
    neutral_test.append(sid.polarity_scores(i)['neu'])
    compound_test.append(sid.polarity_scores(i)['compound'])
```

```
100%|████████████████████████████████████████████████████████████████
██████| 36052/36052 [03:22<00:00, 178.27it/s]
```

In [145]:

```
import numpy as np
neutral_train = np.asarray(neutral_train).reshape(len(preprocessed_essays_Train),1)
positive_train = np.asarray(positive_train).reshape(len(preprocessed_essays_Train),1)
negative_train = np.asarray(negative_train).reshape(len(preprocessed_essays_Train),1)
compound_train = np.asarray(compound_train).reshape(len(preprocessed_essays_Train),1)

print(neutral_train.shape)
print(positive_train.shape)
print(negative_train.shape)
print(compound_train.shape)
```

```
(49041, 1)
(49041, 1)
(49041, 1)
(49041, 1)
```

In [146]:

```
import numpy as np
neutral_cv = np.asarray(neutral_cv).reshape(len(preprocessed_essays_Cv),1)
positive_cv = np.asarray(positive_cv).reshape(len(preprocessed_essays_Cv),1)
negative_cv = np.asarray(negative_cv).reshape(len(preprocessed_essays_Cv),1)
compound_cv = np.asarray(compound_cv).reshape(len(preprocessed_essays_Cv),1)

print(neutral_cv.shape)
print(positive_cv.shape)
print(negative_cv.shape)
print(compound_cv.shape)
```

```
(24155, 1)
(24155, 1)
(24155, 1)
(24155, 1)
```

In [147]:

```
import numpy as np
neutral_test = np.asarray(neutral_test).reshape(len(preprocessed_essays_Test),1)
positive_test = np.asarray(positive_test).reshape(len(preprocessed_essays_Test),1)
negative_test = np.asarray(negative_test).reshape(len(preprocessed_essays_Test),1)
compound_test = np.asarray(compound_test).reshape(len(preprocessed_essays_Test),1)

print(neutral_test.shape)
print(positive_test.shape)
print(negative_test.shape)
print(compound_test.shape)
```

```
(36052, 1)
(36052, 1)
(36052, 1)
(36052, 1)
```

In [148]:

```
type(project_title_word_count_train)
```

Out[148]:

```
numpy.ndarray
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Applying DT(DECISION TREE)

Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

Note that we are using the BOW with bigrams here and min_df value = 10 and max features = 5000 i am mentioning this here specially because trying different values of these will for sure give differeent results and may be better also but will require good computational resources.

Now we need to merge all the numerical vectors(categorical features,text features,numerical features) given above for set-1 which we created using different methods

In [149]:

```
from scipy.sparse import hstack

X_train_merge = hstack((categories_one_hot_train,subcategories_one_hot_train,teacher_prefix_one_hot_train,project_grade_categories_one_hot_train,school_state_categories_one_hot_train,essay_bow_train,project_title_bow_train,price_train,quantity_train,prev_projects_train,project_title_word_count_train,project_essay_word_count_train,neutral_train,positive_train,negative_train,compound_train)).tocsr()
X_test_merge = hstack((categories_one_hot_test,subcategories_one_hot_test,teacher_prefix_one_hot_test,project_grade_categories_one_hot_test,school_state_categories_one_hot_test,essay_bow_test,project_title_bow_test,price_test,quantity_test,prev_projects_test,project_title_word_count_test,project_essay_word_count_test,neutral_test,positive_test,negative_test,compound_test)).tocsr()
X_cv_merge = hstack((categories_one_hot_cv,subcategories_one_hot_cv,teacher_prefix_one_hot_cv,project_grade_categories_one_hot_cv,school_state_categories_one_hot_cv,essay_bow_cv,project_title_bow_cv,price_cv,quantity_cv,prev_projects_cv,project_title_word_count_cv,project_essay_word_count_cv,neutral_cv,positive_cv,negative_cv,compound_cv)).tocsr()
```

In [150]:

```
# this will be our finally created data matrix dimensions

print(X_train_merge.shape, y_train.shape)
print(X_test_merge.shape, y_test.shape)
print(X_cv_merge.shape, y_cv.shape)
```

```
(49041, 16829) (49041,)
(36052, 16829) (36052,)
(24155, 16829) (24155,)
```

Let us just train our Decision tree model based on above data.

In [151]:

```
# refer --> https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTree
Classifier.html

from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier(max_depth = 3)
```

In [152]:

```
clf = decision_tree.fit(X_train_merge,y_train)
```

Now the next important task for us is to visualize this decision tree for that we will need to use the library graphviz

Graphviz requires all the feature names hence we need to extract all the feature names first.

Extracting feature names from set-1 for graphviz

In [288]:

```
set_1_feature_names = []
```

In [289]:

```
sorted_cat_dict
```

Out[289]:

```
{'Warmth': 445,
 'Care_Hunger': 445,
 'History_Civics': 1930,
 'Music_Arts': 3416,
 'AppliedLearning': 4012,
 'SpecialNeeds': 4459,
 'Health_Sports': 4775,
 'Math_Science': 13775,
 'Literacy_Language': 17257}
```

In [290]:

```
# appending the project category features
for i in sorted_cat_dict:
    set_1_feature_names.append(i)
```

In [291]:

```
set_1_feature_names
```

Out[291]:

```
['Warmth',
 'Care_Hunger',
 'History_Civics',
 'Music_Arts',
 'AppliedLearning',
 'SpecialNeeds',
 'Health_Sports',
 'Math_Science',
 'Literacy_Language']
```

In [292]:

```
# appending the project sub category features
for i in sorted_sub_cat_dict:
    set_1_feature_names.append(i)
```

In [293]:

```
len(set_1_feature_names)
```

Out[293]:

```
39
```

In [294]:

```
# appending the teacher prefix features
for i in mylist_teacher_prefix_actual_Train:
    set_1_feature_names.append(i)
```

In [295]:

```
len(set_1_feature_names)
```

Out[295]:

```
44
```

In [296]:

```
# appending the project grade category features
for i in mylist_project_grade_category_actual:
    set_1_feature_names.append(i)
```

In [297]:

```
len(set_1_feature_names)
```

Out[297]:

```
48
```

In [298]:

```
# appending the school states features
for i in mylist_actual:
    set_1_feature_names.append(i)
```

In [299]:

```
len(set_1_feature_names)
```

Out[299]:

99

In [300]:

```
# appending the features received while applying bow on project essays
for i in vectorizer_project_essay_bow.get_feature_names():
    set_1_feature_names.append(i)
```

In [301]:

```
len(set_1_feature_names)
```

Out[301]:

5099

In [302]:

```
# appending the features received while applying bow on project essays title
for i in vectorizer_project_title_bow.get_feature_names():
    set_1_feature_names.append(i)
```

In [303]:

```
len(set_1_feature_names)
```

Out[303]:

16820

Now appending the remianing feature names which are in our train data for set 1

In [168]:

```
set_1_feature_names.append("project_essay_title_word_count")
set_1_feature_names.append("project_essay_word_count")
set_1_feature_names.append("positive")
set_1_feature_names.append("neutral")
set_1_feature_names.append("negative")
set_1_feature_names.append("compound")
set_1_feature_names.append("price")
set_1_feature_names.append("quantity")
set_1_feature_names.append("number_of_previous_projects")
```

In [169]:

```
len(set_1_feature_names)
```

Out[169]:

```
16829
```

In [170]:

```
print(X_train_merge.shape, y_train.shape)
print(X_test_merge.shape, y_test.shape)
print(X_cv_merge.shape, y_cv.shape)
```

```
(49041, 16829) (49041,)
(36052, 16829) (36052,)
(24155, 16829) (24155,)
```

Hence we can clearly see above that all the feature names have been added to the created list as 16811 matches with both.

Using graphviz to visualize the decision tree

In [171]:

```
!pip install pydot
```

```
Requirement already satisfied: pydot in c:\users\rashu tyagi\anaconda3\lib
\site-packages (1.4.1)
Requirement already satisfied: pyparsing>=2.1.4 in c:\users\rashu tyagi\an
aconda3\lib\site-packages (from pydot) (2.3.1)
```

In [172]:

```
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydot
import pydotplus
from sklearn import tree
import graphviz
```

In [173]:

```
!pip install pydotplus
```

```
Requirement already satisfied: pydotplus in c:\users\rashu tyagi\anaconda3
\lib\site-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\rashu tyagi\an
aconda3\lib\site-packages (from pydotplus) (2.3.1)
```

In [174]:

```
!pip install graphviz
```

```
Requirement already satisfied: graphviz in c:\users\rashu tyagi\anaconda3
\lib\site-packages (0.11.1)
```

In [175]:

```
# refer -- > https://chrisalbon.com/machine_learning/trees_and_forests/visualize_a_decision_tree/
# refer ---> https://www.youtube.com/watch?v=XxJKt9s0Dlg

dot_data = tree.export_graphviz(decision_tree, out_file=None,feature_names=set_1_feature_names)

graph = graphviz.Source(dot_data)
graph.format = 'png'
graph.render("Set -1 Tree with BOW features",view = True)
#graph = pydotplus.graph_from_dot_data(dot_data)

#Image(graph.create_png())
```

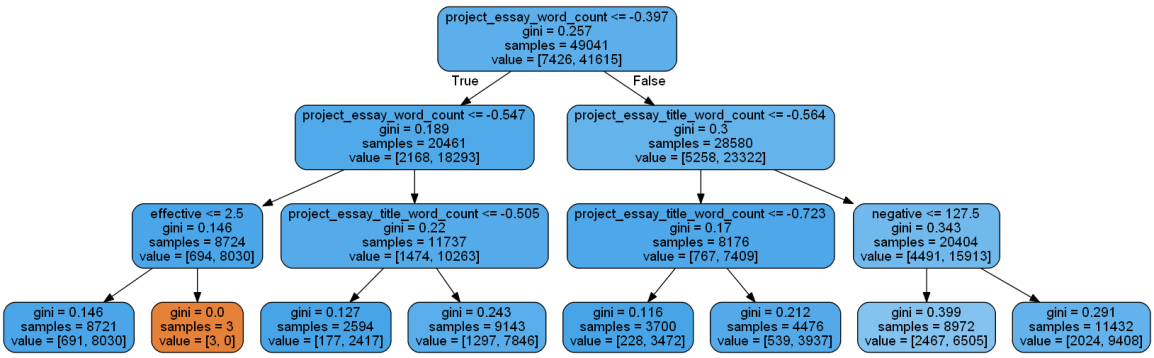
Out[175]:

```
'Set -1 Tree with BOW features.png'
```

In [176]:

```
dot_data = tree.export_graphviz(decision_tree, out_file=None,feature_names=set_1_feature_names,filled=True,rounded=True)

graph = pydotplus.graph_from_dot_data(dot_data)

Image(graph.create_png())
```

Out[176]:



Now the Important task which comes is to know what best hyperparameters should we pass to the decision tree classifier to get the best results in terms of accuracy.

For the task of getting the best hyperparameters we will be doing hyperparameter tuning and as we know that we have two hyperparameters in case of decision tree and they are 1.) Maximum depth of the decision tree and 2.) minimum number of sample split

Minimum number of sample split means what should be the minimum number of data points present in a node in order to split the node.

Using GridSearchCV for the purpose of hyperparameter tuning.

Importing the required modules

In [177]:

```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Machine%20Learning%20Lecture%202.html

from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
```

Applying the GridsearchCV to find the best value of hyperparameters

In [178]:

```
decision_tree = DecisionTreeClassifier()

tuned_parameters = {'max_depth':[1,5,10,100,500], 'min_samples_split': [5,10,50,100,500]}
```

In [179]:

```
# refer --> https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
# cv = 10 means 10 fold crossvalidation

clf = GridSearchCV(decision_tree,tuned_parameters,cv=10,scoring = 'roc_auc',n_jobs=-1,verbose=10)

clf.fit(X_train_merge,y_train)
```

```
Fitting 10 folds for each of 25 candidates, totalling 250 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 6 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    3.1s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    4.3s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:    5.5s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:    6.8s
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:    9.2s
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:   11.3s
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:   14.0s
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:   19.4s
[Parallel(n_jobs=-1)]: Done  73 tasks      | elapsed:   25.9s
[Parallel(n_jobs=-1)]: Done  86 tasks      | elapsed:   31.3s
[Parallel(n_jobs=-1)]: Done 101 tasks      | elapsed:   44.4s
[Parallel(n_jobs=-1)]: Done 116 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 133 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done 150 tasks      | elapsed:  1.8min
[Parallel(n_jobs=-1)]: Done 169 tasks      | elapsed:  7.9min
[Parallel(n_jobs=-1)]: Done 188 tasks      | elapsed: 12.2min
[Parallel(n_jobs=-1)]: Done 209 tasks      | elapsed: 17.6min
[Parallel(n_jobs=-1)]: Done 230 tasks      | elapsed: 23.4min
[Parallel(n_jobs=-1)]: Done 250 out of 250 | elapsed: 28.3min finished
```

Out[179]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'max_depth': [1, 5, 10, 100, 500], 'min_samples_split': [5, 10, 50, 100, 500]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=10)
```

In [180]:

```
#https://stackoverflow.com/questions/44947574/what-is-the-meaning-of-mean-test-score-in-cv-result

train_auc= clf.cv_results_['mean_train_score']
cv_auc = clf.cv_results_['mean_test_score']
```

Now we have 3 things to compare 1.) we need to compare our parameters max_depth and min_samples_split with our train_auc score on a single graph 2.) we need to compare our parameters max_depth and min_samples_split with our cv_auc score on a single graph

In order to have all 3 things on the same graph what i will be using here is the heatmaps representations.

Comparing for train_auc

In [181]:

```
# importing the required libraries

import numpy as np
import seaborn as sns
```

In [182]:

```
max_depth = [1,5,10,100,500]
min_samples_split = [5,10,50,100,500]
```

In [183]:

```
train_auc.shape
```

Out[183]:

```
(25,)
```

In [184]:

```
cv_auc.shape
```

Out[184]:

```
(25,)
```

In [185]:

```
# creating dataframe to plot heatmap for train_auc

#train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Sp
lit':min_samples_split, 'Train_AUC_Score':train_auc})

# using the above code I was getting error that all arrays must be of same size and thi
s happened because max_depth and min_samples_split are of size 5 each only
# while train_auc has size of 25

# now in order to deal with it I will keep the value of max_depth five times each so wi
th mij samples split because the algorithm of
# grid search cv works this way only that it compapres one with all elements of opther
 parameter list and so on.
```

In [186]:

```
max_depth = [1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,100,100,100,100,100,500,500,500,500,500
]
min_samples_split = [5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,
10,50,100,500]
```

In [187]:

```
len(max_depth)
```

Out[187]:

25

In [188]:

```
len(min_samples_split)
```

Out[188]:

25

In [189]:

```
# creating dataframe to plot heatmap for train_auc

train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Spl
it':min_samples_split, 'Train_AUC_Score':train_auc})
```

In [190]:

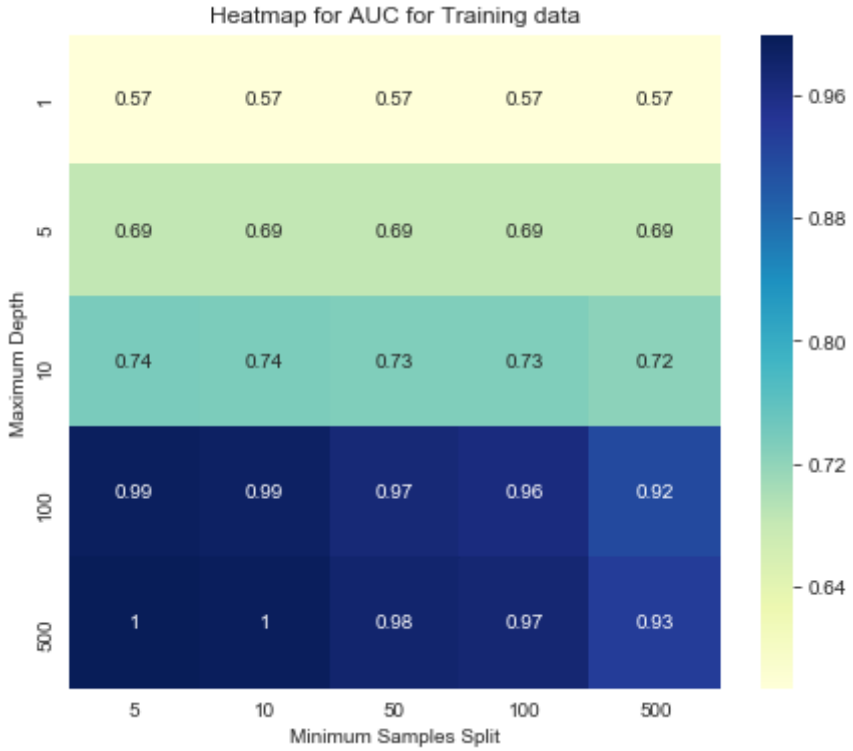
```
# adding the headings of every dimesion
train_auc_dataframe = train_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum
Samples Split', values='Train_AUC_Score')
```

In [191]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(train_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC
 for Training data')
```

Out[191]:

```
Text(0.5, 1.0, 'Heatmap for AUC for Training data')
```



Comparing for cv_auc

In [192]:

```
# creating dataframe to plot heatmap for cv_auc

cv_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Split'
:min_samples_split, 'CV_AUC_Score':cv_auc})
```

In [193]:

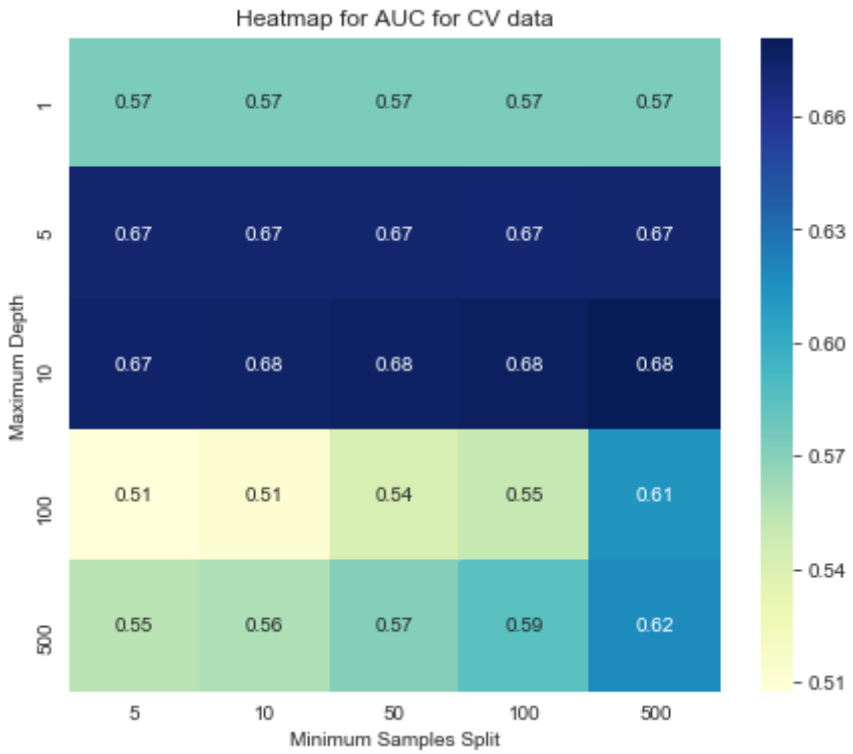
```
# adding the headings of every dimension
cv_auc_dataframe = cv_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum Samples Split', values='CV_AUC_Score')
```

In [194]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(cv_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC for CV data')
```

Out[194]:

```
Text(0.5, 1.0, 'Heatmap for AUC for CV data')
```



Now seeing the two heatmaps from above the best value of max_depth is 10 and min_sample_split is 100

Training our final model based on the value of parameters received from above

In [195]:

```
from sklearn.tree import DecisionTreeClassifier

model_decision_tree = DecisionTreeClassifier(max_depth = 10, min_samples_split = 100)

model_decision_tree.fit(X_train_merge,y_train)
```

Out[195]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=10,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=100,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best')
```

In [196]:

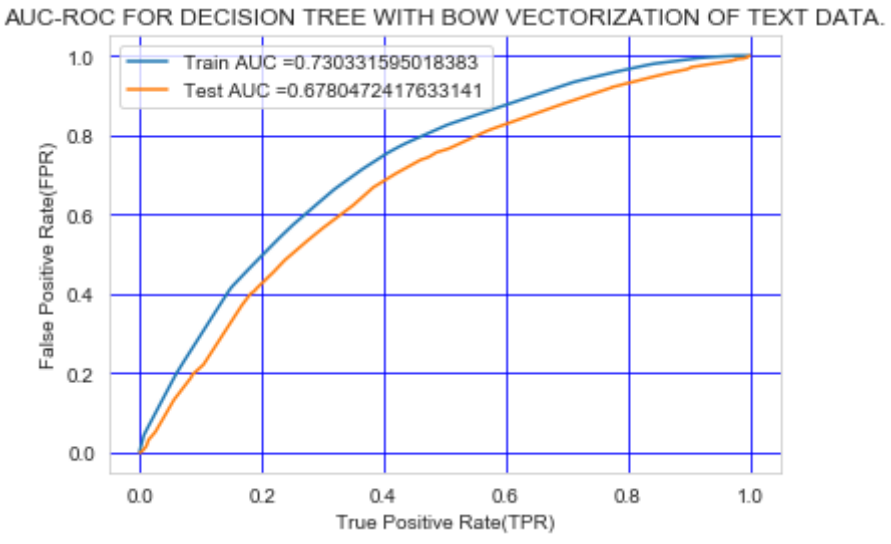
```
y_train_pred =  model_decision_tree.predict_proba(X_train_merge)[:,1]

y_test_pred =  model_decision_tree.predict_proba(X_test_merge)[:,1]
```

In [197]:

```
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [198]:

```
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.grid(b=True, which='major', color='b', linestyle='-')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC-ROC FOR DECISION TREE WITH BOW VECTORIZATION OF TEXT DATA.")

plt.show()
```



We received the train accuracy of 0.73 and test accuracy of 0.69 which is not that bad actually.

Plotting the Decision tree for set 1 using best parameters using graphviz

In [199]:

```
# refer -- > https://chrisalbon.com/machine_learning/trees_and_forests/visualize_a_decision_tree/
# refer ---> https://www.youtube.com/watch?v=XxJKt9s0Dlg

dot_data = tree.export_graphviz(model_decision_tree, out_file=None,feature_names=set_1_feature_names)

graph = graphviz.Source(dot_data)
graph.format = 'png'
graph.render("Set -1 Tree with BOW features with best parameters values found using gridsearch",view = True)
#graph = pydotplus.graph_from_dot_data(dot_data)

#Image(graph.create_png())
```

Out[199]:

```
'Set -1 Tree with BOW features with best parameters values found using gri
dsearch.png'
```

In [200]:

```
dot_data = tree.export_graphviz(model_decision_tree, out_file=None,feature_names=set_1_feature_names,filled=True,rounded=True)

graph = pydotplus.graph_from_dot_data(dot_data)

Image(graph.create_png())
```

Out[200]:

I tried many things but could not make it visible clearly due to large number of depth.

Confusion matrix for above data

In [201]:

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

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

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

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

Confusion matrix for train and test data for BOW vectorization

In [202]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999200297047033 for threshold 0.793
[[ 3692  3734]
 [ 7285 34330]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24996234146461282 for threshold 0.813
[[ 2696  2763]
 [ 7172 23421]]
```

Visually plotting the confusion matrix for training data

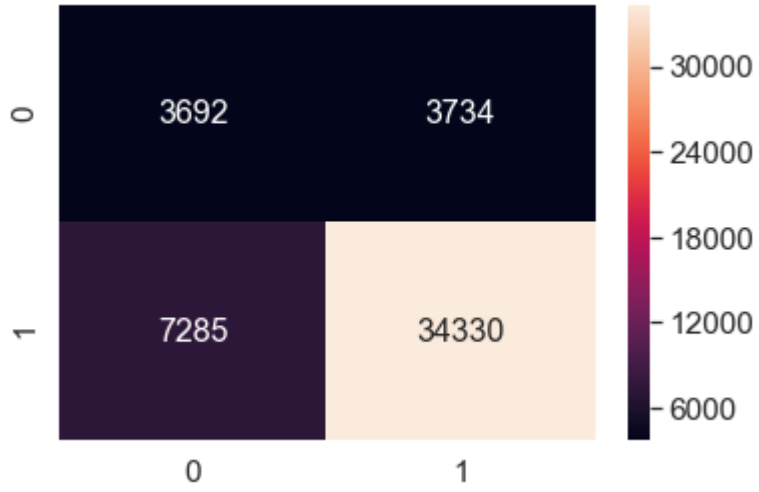
In [203]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt


df_cm = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred, tr_thresholds, trai
n_fpr, train_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24999200297047033 for threshold 0.793
```

Out[203]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc1c0dd710>
```



Visually plotting the confusion matrix for test data

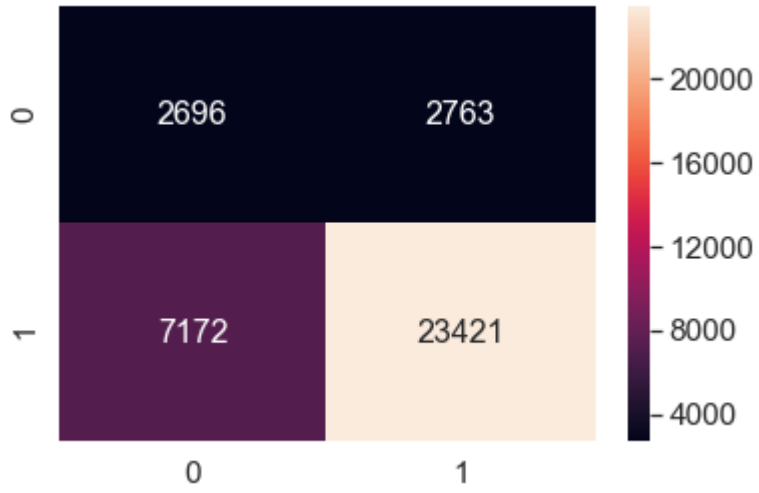
In [204]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_test = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred, tr_thresholds, t
est_fpr, test_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_test, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24996234146461282 for threshold 0.813
```

Out[204]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc16be4828>
```



Analysis of False Positive Test Points On Our Essay Text.

In [205]:

```
essay_bow_test.shape
```

Out[205]:

```
(36052, 5000)
```

In [206]:

```
type(essay_bow_test)
```

Out[206]:

```
scipy.sparse.csr.csr_matrix
```

Now out of all the 36052 test daya points for essay we need to find out those which were declared as False Positives.

False Positive -->> Label which was predicted as Positive, but is actually Negative.

In [207]:

```
X_test_merge.shape
```

Out[207]:

```
(36052, 16829)
```

In [208]:

```
# getting all the words for which we have performed the vectorization
bow_words_vectorized = vectorizer_project_essay_bow.get_feature_names()
```

In [209]:

```
bow_words_vectorized[0:20]
```

Out[209]:

```
['000',
 '10',
 '100',
 '100 free',
 '100 percent',
 '100 students',
 '11',
 '12',
 '12th',
 '13',
 '14',
 '15',
 '16',
 '17',
 '18',
 '19',
 '1st',
 '1st grade',
 '20',
 '20 students']
```

In [210]:

```
# number of words vectorized
len(bow_words_vectorized)
```

Out[210]:

```
5000
```

Now we have the predicted values in the y_test_pred variable

In [211]:

```
type(y_test_pred)
```

Out[211]:

```
numpy.ndarray
```

In [212]:

```
y_test_pred.shape
```

Out[212]:

```
(36052,)
```

In [213]:

```
y_test_pred[0:20]
```

Out[213]:

```
array([0.87994249, 0.9424226 , 0.84712329, 0.51694095, 0.89803094,
       0.94199444, 0.74283194, 0.64909091, 0.74549098, 0.96944444,
       0.87994249, 0.8970013 , 0.84712329, 0.88140351, 0.88140351,
       0.9109589 , 0.93114311, 0.89803094, 0.93114311, 0.88140351])
```

Now the above values are the predicted probability values of the data points whether they will belong to the positive class or the negative class.In the confusion matrix code starting part we can see the variable t which gives us the threshold value which means the value above which a data point is considered in the positive class.

For the test data it can be directly seen from confusion matrix code that threshold value = 0.804

We want all those points which have predict_proba values greater than 0.804 because they are the predicted positive points.

Along with them we will need data points whose actual value(y_test) = 0 because they will be the points whose actual values will be negative.

And after combining the above two conditions we will reach to the points whic were predicted to be positive but are actually negative which is actually the False positive points.

In [214]:

```
type(y_test)
```

Out[214]:

```
pandas.core.series.Series
```

In [215]:

```
y_test_list = list(y_test)
```

In [216]:

```
len(y_test_list)
```

Out[216]:

```
36052
```

In [217]:

```
threshold_value_test = 0.804
```

We will try to get the index of all those points which are false positives.

In [218]:

```
fp_index = []
number_of_fp = 0
k = len(y_test_list)
for j in tqdm(range(k)):
    if y_test_list[j] == 0 and y_test_pred[j] >= threshold_value_test:
        fp_index.append(j)
        number_of_fp = number_of_fp + 1
```

```
100%|██████████████████████████████████████████████████████████████████████
██| 36052/36052 [00:00<00:00, 2780980.76it/s]
```

In [219]:

```
fp_index[2590:2605]
```

Out[219]:

```
[34040,
 34049,
 34060,
 34066,
 34067,
 34071,
 34073,
 34077,
 34081,
 34083,
 34091,
 34098,
 34116,
 34129,
 34139]
```

In [220]:

```
len(fp_index)
```

Out[220]:

```
2763
```

So there are 2605 false positive points and yes they are different from the False positive shown in confusion matrix and this happened because the model was trained again for essay text and there is just a very small difference of value so no worries.

Now I want the words at above indexes only

In [221]:

```
len(bow_words_vectorized)
```

Out[221]:

```
5000
```

In [222]:

```
final_fp_words = []
for k in fp_index:
    if k<5000:
        final_fp_words.append(str(bow_words_vectorized[k]))
```

In [223]:

```
len(final_fp_words)
```

Out[223]:

```
367
```

In [224]:

```
final_fp_words[0:20]
```

Out[224]:

```
['15',
 '24',
 '3d printer',
 '45',
 '5th graders',
 '8th',
 '90',
 '99',
 'able',
 'able explore',
 'able use',
 'academic',
 'access books',
 'accomplishments',
 'accountable',
 'achieving',
 'activities',
 'age',
 'age appropriate',
 'allowing students']
```

In [225]:

```
final_fp_words == bow_words_vectorized
```

Out[225]:

```
False
```

Creating Word Cloud for the False Positive Words From Essay

In [226]:

```
!pip install wordcloud
```

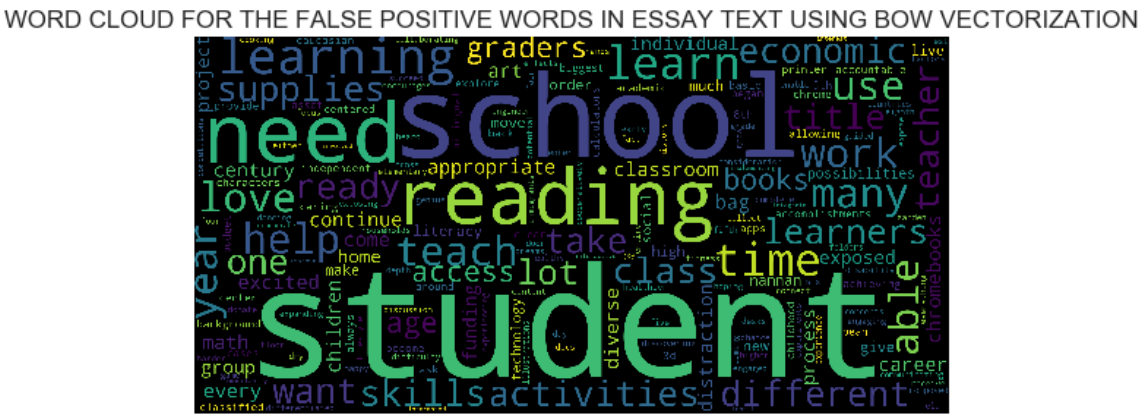
```
Requirement already satisfied: wordcloud in c:\users\rashu tyagi\anaconda3
\lib\site-packages (1.5.0)
Requirement already satisfied: pillow in c:\users\rashu tyagi\anaconda3\li
b\site-packages (from wordcloud) (5.4.1)
Requirement already satisfied: numpy>=1.6.1 in c:\users\rashu tyagi\anacon
da3\lib\site-packages (from wordcloud) (1.16.2)
```

In [227]:

```
# refer ---> https://www.geeksforgeeks.org/generating-word-cloud-python/

# importing all necessery modules
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

stopwords = set(STOPWORDS)

#NOTE THAT WE NEED A SINGLE STRING WITH ALL THE WORDS SEPARATED BY SPACE WHOSE WORD CLO
UD WEWANT TO CREATE.

final_word_string = ' '
for words in final_fp_words:
    final_word_string = final_word_string + words + " "



wordcloud = WordCloud(width = 2000, height = 1000,background_color ='black',stopwords =
stopwords,min_font_size = 10).generate(final_word_string)
```

In [228]:

```
# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.title("WORD CLOUD FOR THE FALSE POSITIVE WORDS IN ESSAY TEXT USING BOW VECTORIZATIO
N")

plt.show()
```



Box-Plot for these False Positive Data Points.

For drawing the box plot we will need a dataframe with all these false positive data points

In [229]:

```
box_plot_dataframe_price = pd.DataFrame(X_test['price'])
```

In [230]:

```
# keeping only those indices which are refered as the false positves and removing all o
ther
box_plot_dataframe_price = box_plot_dataframe_price.iloc[fp_index,:]
```

In [231]:

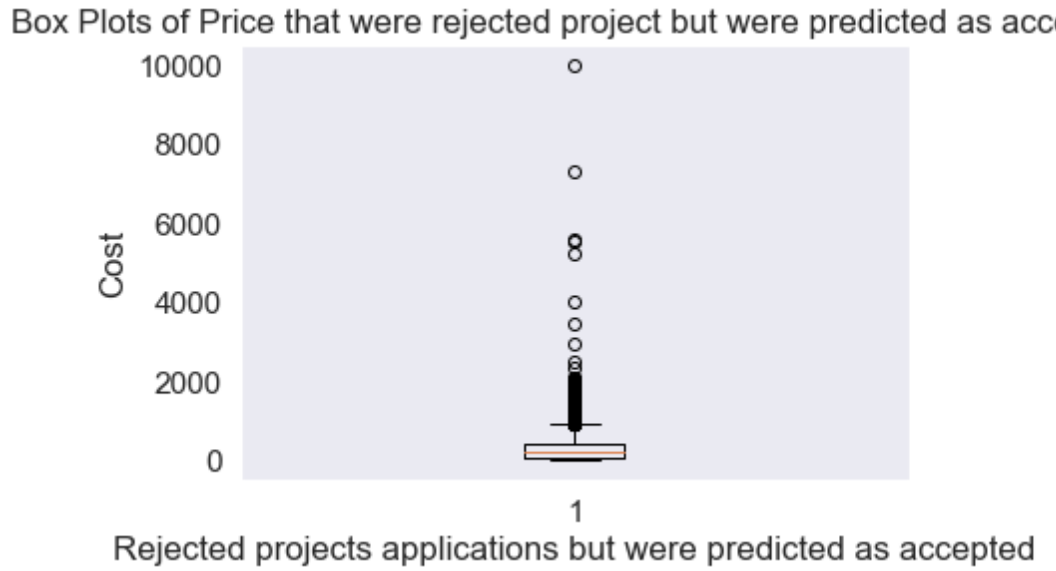
```
box_plot_dataframe_price.shape
```

Out[231]:

```
(2763, 1)
```

In [232]:

```
# creating the box plot

plt.boxplot(box_plot_dataframe_price.values)
plt.title('Box Plots of Price that were rejected project but were predicted as accepte
d')
plt.xlabel('Rejected projects applications but were predicted as accepted')
plt.ylabel('Cost')
plt.grid()
plt.show()
```



From above we can see that most of the wrongly accepted projects are costing very less at around 300 they might be accepted wrongly due to less cost included but actually they were rejected

PDF with the Teacher_number_of_previously_posted_projects of the False Positive data points.

In [233]:

```
pdf_dataframe_teacher_number_of_previously_posted_projects = pd.DataFrame(X_test['teach
er_number_of_previously_posted_projects'])
```

In [234]:

```
# keeping only those indices which are refered as the false positves and removing all o
ther
pdf_dataframe_teacher_number_of_previously_posted_projects = pdf_dataframe_teacher_numb
er_of_previously_posted_projects.iloc[fp_index,:]
```

In [235]:

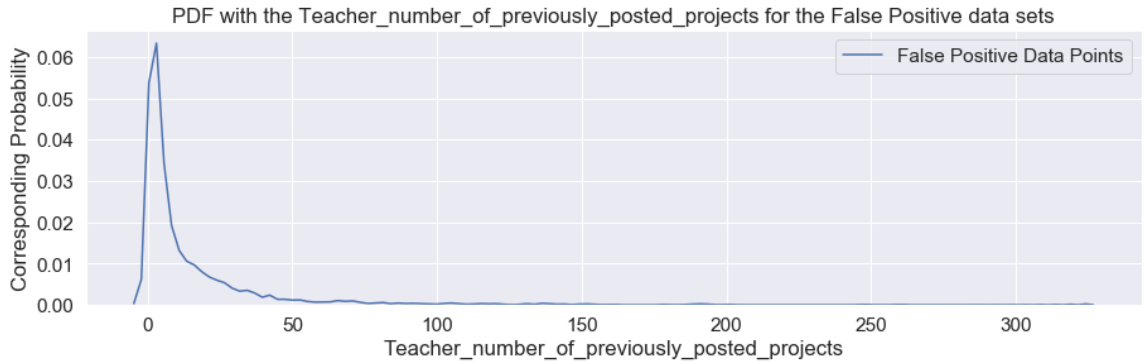
```
pdf_dataframe_teacher_number_of_previously_posted_projects.shape
```

Out[235]:

```
(2763, 1)
```

In [236]:

```
# plotting the pdf

plt.figure(figsize=(15,4))
sns.distplot(pdf_dataframe_teacher_number_of_previously_posted_projects.values, hist=Fa
lse, label="False Positive Data Points")
plt.title('PDF with the Teacher_number_of_previously_posted_projects for the False Posi
tive data sets')
plt.xlabel('Teacher_number_of_previously_posted_projects')
plt.ylabel('Corresponding Probability')
plt.legend()
plt.show()
```



From above we can say that most of the projects which were predicted as accpeted but were actually rejected have number of previously posted projects by a teacher in the range of 0-10 only.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Set 2 : categorical, numerical features + project_title(TFIDF) + preprocessed_essay (TFIDF)

Note that we are using the tfidf with bigrams here and min_df value = 10 and max features = 5000 i am mentioning this here specially because trying different values of these will for sure give differeent results and may be better also but will require good computational resources.

Now we need to merge all the numerical vectors(categorical features,text features,numerical features) given above for set-2 which we created using different methods

In [196]:

```
from scipy.sparse import hstack

X_train_merge_set_2 = hstack((categories_one_hot_train,subcategories_one_hot_train,teac
her_prefix_one_hot_train,project_grade_categories_one_hot_train,school_state_categories
_one_hot_train,text_tfidf_train,project_title_tfidf_train,price_train,quantity_train,pr
ev_projects_train,project_title_word_count_train,project_essay_word_count_train,neutral
_train,positive_train,negative_train,compound_train)).tocsr()
X_test_merge_set_2 = hstack((categories_one_hot_test,subcategories_one_hot_test,teacher
_prefix_one_hot_test,project_grade_categories_one_hot_test,school_state_categories_one_
hot_test,text_tfidf_test,project_title_tfidf_test,price_test,quantity_test,prev_project
s_test,project_title_word_count_test,project_essay_word_count_test,neutral_test,positiv
e_test,negative_test,compound_test)).tocsr()
X_cv_merge_set_2 = hstack((categories_one_hot_cv,subcategories_one_hot_cv,teacher_prefi
x_one_hot_cv,project_grade_categories_one_hot_cv,school_state_categories_one_hot_cv,tex
t_tfidf_cv,title_tfidf_cv,price_cv,quantity_cv,prev_projects_cv,project_title_word_coun
t_cv,project_essay_word_count_cv,neutral_cv,positive_cv,negative_cv,compound_cv)).tocsr
()
```

In [197]:

```
# this will be our finally created data matrix dimensions

print(X_train_merge_set_2.shape, y_train.shape)
print(X_test_merge_set_2.shape, y_test.shape)
print(X_cv_merge_set_2.shape, y_cv.shape)
```

```
(49041, 8108) (49041,)
(36052, 8108) (36052,)
(24155, 8108) (24155,)
```

Let us just train our Decision tree model based on above data.

In [198]:

```
# refer --> https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTree
Classifier.html

from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier(max_depth = 3)
```

In [199]:

```
clf = decision_tree.fit(X_train_merge_set_2,y_train)
```

Now the next important task for us is to visualize this decision tree for that we will need to use the library graphviz

Graphviz requires all the feature names hence we need to extract all the feature names first.

Extracting feature names from essay on set-2 for graphviz

In [200]:

```
set_2_feature_names = []
```

In [201]:

```
# appending the project category features
for i in sorted_cat_dict:
    set_2_feature_names.append(i)
```

In [202]:

```
len(set_2_feature_names)
```

Out[202]:

9

In [203]:

```
# appending the project sub category features
for i in sorted_sub_cat_dict:
    set_2_feature_names.append(i)
```

In [204]:

```
len(set_2_feature_names)
```

Out[204]:

39

In [205]:

```
# appending the teacher prefix features
for i in mylist_teacher_prefix_actual_Train:
    set_2_feature_names.append(i)
```

In [206]:

```
len(set_2_feature_names)
```

Out[206]:

44

In [207]:

```
# appending the project grade category features
for i in mylist_project_grade_category_actual:
    set_2_feature_names.append(i)
```

In [208]:

```
len(set_2_feature_names)
```

Out[208]:

48

In [209]:

```
# appending the school states features
for i in mylist_actual:
    set_2_feature_names.append(i)
```

In [210]:

```
len(set_2_feature_names)
```

Out[210]:

99

In [211]:

```
#tfidf_words
```

In [212]:

```
type(tfidf_words)
```

Out[212]:

set

In [213]:

```
tfidf_words_list = list(tfidf_words)
```

In [214]:

```
type(tfidf_words_list)
```

Out[214]:

list

In [215]:

```
# appending the features received while applying tfidf on project essays
for i in tfidf_words_list:
    set_2_feature_names.append(i)
```

In [216]:

```
len(set_2_feature_names)
```

Out[216]:

5099

In [217]:

```
#tfidf_project_title_words
```

In [218]:

```
type(tfidf_project_title_words)
```

Out[218]:

```
set
```

In [219]:

```
tfidf_project_title_words_list = list(tfidf_project_title_words)
```

In [220]:

```
type(tfidf_project_title_words_list)
```

Out[220]:

```
list
```

In [221]:

```
# appending the features received while applying tfidf on project titles
for i in tfidf_project_title_words_list:
    set_2_feature_names.append(i)
```

In [222]:

```
len(set_2_feature_names)
```

Out[222]:

```
8099
```

Now appending the remianing feature names which are in our train data for set 2

In [223]:

```
set_2_feature_names.append("project_essay_title_word_count")
set_2_feature_names.append("project_essay_word_count")
set_2_feature_names.append("positive")
set_2_feature_names.append("neutral")
set_2_feature_names.append("negative")
set_2_feature_names.append("compound")
set_2_feature_names.append("price")
set_2_feature_names.append("quantity")
set_2_feature_names.append("number_of_previous_projects")
```

In [224]:

```
len(set_2_feature_names)
```

Out[224]:

```
8108
```

In [225]:

```
print(X_train_merge_set_2.shape, y_train.shape)
print(X_test_merge_set_2.shape, y_test.shape)
print(X_cv_merge_set_2.shape, y_cv.shape)
```

```
(49041, 8108) (49041,)
(36052, 8108) (36052,)
(24155, 8108) (24155,)
```

Hence we can clearly see above that all the feature names have been added to the created list as 8108 matches with both.

Using graphviz to visualize the decision tree

In [330]:

```
!pip install pydot
```

```
Requirement already satisfied: pydot in c:\users\rashu tyagi\anaconda3\lib
\site-packages (1.4.1)
Requirement already satisfied: pyparsing>=2.1.4 in c:\users\rashu tyagi\an
aconda3\lib\site-packages (from pydot) (2.3.1)
```

In [331]:

```
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydot
import pydotplus
from sklearn import tree
import graphviz
```

In [332]:

```
!pip install pydotplus
```

```
Requirement already satisfied: pydotplus in c:\users\rashu tyagi\anaconda3
\lib\site-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\rashu tyagi\an
aconda3\lib\site-packages (from pydotplus) (2.3.1)
```

In [333]:

```
!pip install graphviz
```

```
Requirement already satisfied: graphviz in c:\users\rashu tyagi\anaconda3
\lib\site-packages (0.11.1)
```

In [334]:

```
# refer -- > https://chrisalbon.com/machine_learning/trees_and_forests/visualize_a_decision_tree/
# refer ---> https://www.youtube.com/watch?v=XxJKt9s0Dlg

dot_data = tree.export_graphviz(decision_tree, out_file=None,feature_names=set_2_feature_names)

graph = graphviz.Source(dot_data)
graph.format = 'png'
graph.render("Set -2 Tree with TFIDF features",view = True)
#graph = pydotplus.graph_from_dot_data(dot_data)

#Image(graph.create_png())
```

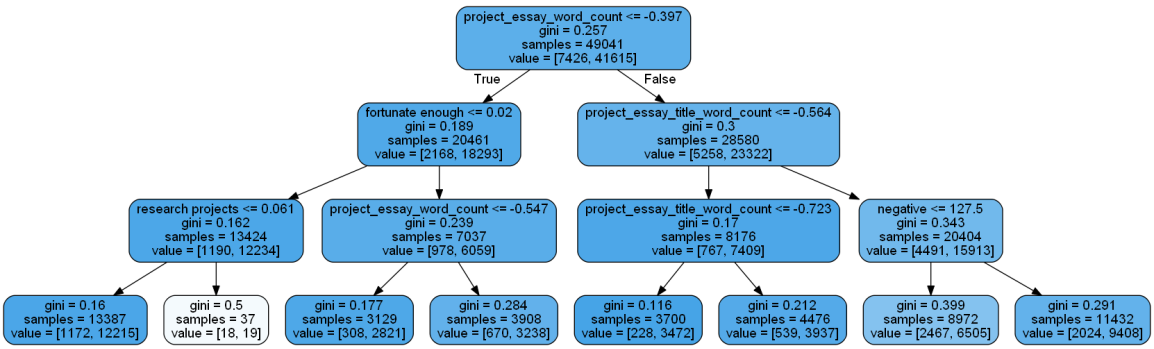
Out[334]:

```
'Set -2 Tree with TFIDF features.png'
```

In [335]:

```
dot_data = tree.export_graphviz(decision_tree, out_file=None,feature_names=set_2_feature_names,filled=True,rounded=True)

graph = pydotplus.graph_from_dot_data(dot_data)

Image(graph.create_png())
```

Out[335]:



Now the Important task which comes is to know what best hyperparameters should we pass to the decision tree classifier to get the best results in terms of accuracy.

For the task of getting the best hyperparameters we will be doing hyperparameter tuning and as we know that we have two hyperparameters in case of decision tree and they are 1.) Maximum depth of the decision tree and 2.) minimum number of sample split

Minimum number of sample split means what should be the minimum number of data points present in a node in order to split the node.

Importing the required modules

In [336]:

```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Machine%20Learning%20Lecture%202.html

from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
```

Applying the GridsearchCV to find the best value of hyperparameters

In [337]:

```
decision_tree = DecisionTreeClassifier()

tuned_parameters = {'max_depth':[1,5,10,100,500], 'min_samples_split': [5,10,50,100,500]}
```

In [338]:

```
# refer --> https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
# cv = 10 means 10 fold crossvalidation

clf = GridSearchCV(decision_tree,tuned_parameters,cv=10,scoring = 'roc_auc',n_jobs=-1,verbose=10)

clf.fit(X_train_merge_set_2,y_train)
```

```
Fitting 10 folds for each of 25 candidates, totalling 250 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 6 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    2.6s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    3.5s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:    5.1s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:    6.7s
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:    8.9s
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:   11.4s
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:   13.5s
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:   26.1s
[Parallel(n_jobs=-1)]: Done  73 tasks      | elapsed:   36.5s
[Parallel(n_jobs=-1)]: Done  86 tasks      | elapsed:   46.5s
[Parallel(n_jobs=-1)]: Done 101 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done 116 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done 133 tasks      | elapsed:  2.2min
[Parallel(n_jobs=-1)]: Done 150 tasks      | elapsed:  2.9min
[Parallel(n_jobs=-1)]: Done 169 tasks      | elapsed:  9.9min
[Parallel(n_jobs=-1)]: Done 188 tasks      | elapsed: 15.4min
[Parallel(n_jobs=-1)]: Done 209 tasks      | elapsed: 22.9min
[Parallel(n_jobs=-1)]: Done 230 tasks      | elapsed: 30.2min
[Parallel(n_jobs=-1)]: Done 250 out of 250 | elapsed: 37.0min finished
```

Out[338]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'max_depth': [1, 5, 10, 100, 500], 'min_samples_split': [5, 10, 50, 100, 500]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=10)
```

In [339]:

```
#https://stackoverflow.com/questions/44947574/what-is-the-meaning-of-mean-test-score-in-cv-result

train_auc= clf.cv_results_['mean_train_score']
cv_auc = clf.cv_results_['mean_test_score']
```

Now we have 3 things to compare 1.) we need to compare our parameters max_depth and min_samples_split with our train_auc score on a single graph 2.) we need to compare our parameters max_depth and min_samples_split with our cv_auc score on a single graph

In order to have all 3 things on the same graph what i will be using here is the heatmaps representations.

Comparing for train_auc

In [340]:

```
# importing the required libraries

import numpy as np
import seaborn as sns
```

In [341]:

```
max_depth = [1,5,10,100,500]
min_samples_split = [5,10,50,100,500]
```

In [342]:

```
train_auc.shape
```

Out[342]:

```
(25,)
```

In [343]:

```
cv_auc.shape
```

Out[343]:

```
(25,)
```

In [345]:

```
# creating dataframe to plot heatmap for train_auc

#train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Sp
lit':min_samples_split, 'Train_AUC_Score':train_auc})

# using the above code I was getting error that all arrays must be of same size and thi
s happened because max_depth and min_samples_split are of size 5 each only
# while train_auc has size of 25

# now in order to deal with it I will keep the value of max_depth five times each so wi
th mij samples split because the algorithm of
# grid search cv works this way only that it compapres one with all elements of opther
 parameter list and so on.


# WE CAN DO IT VICE VERSA ALSO LIKE WRITING 1,5,10,100,500 5 TIMES FOR MAX DEPTH AND WR
ITING 55555,10,10,10,10,10 SO ON FOR MIN_SAMPLES_SPLIT
```

In [346]:

```
max_depth = [1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,100,100,100,100,100,500,500,500,500,500
]
min_samples_split = [5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,
10,50,100,500]
```

In [347]:

```
len(max_depth)
```

Out[347]:

25

In [348]:

```
len(min_samples_split)
```

Out[348]:

25

In [349]:

```
# creating dataframe to plot heatmap for train_auc

train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Spl
it':min_samples_split, 'Train_AUC_Score':train_auc})
```

In [350]:

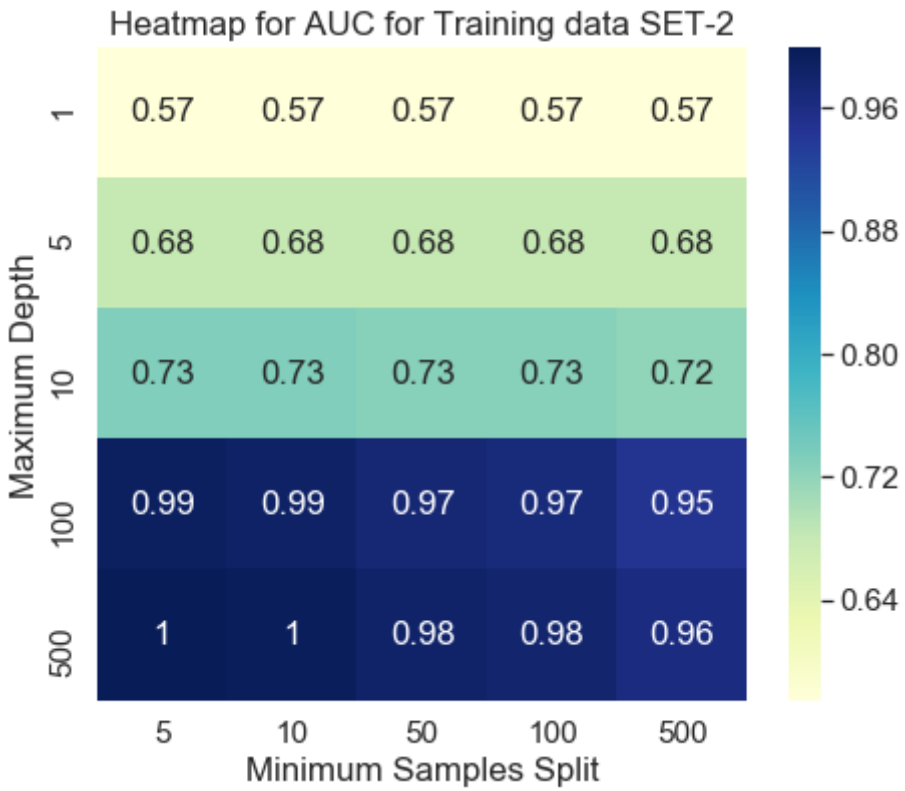
```
# adding the headings of every dimesion
train_auc_dataframe = train_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum
Samples Split', values='Train_AUC_Score')
```

In [352]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(train_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC
 for Training data SET-2')
```

Out[352]:

```
Text(0.5, 1.0, 'Heatmap for AUC for Training data SET-2')
```



Comparing for cv_auc

In [353]:

```
# creating dataframe to plot heatmap for cv_auc

cv_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Split'
:min_samples_split, 'CV_AUC_Score':cv_auc})
```

In [354]:

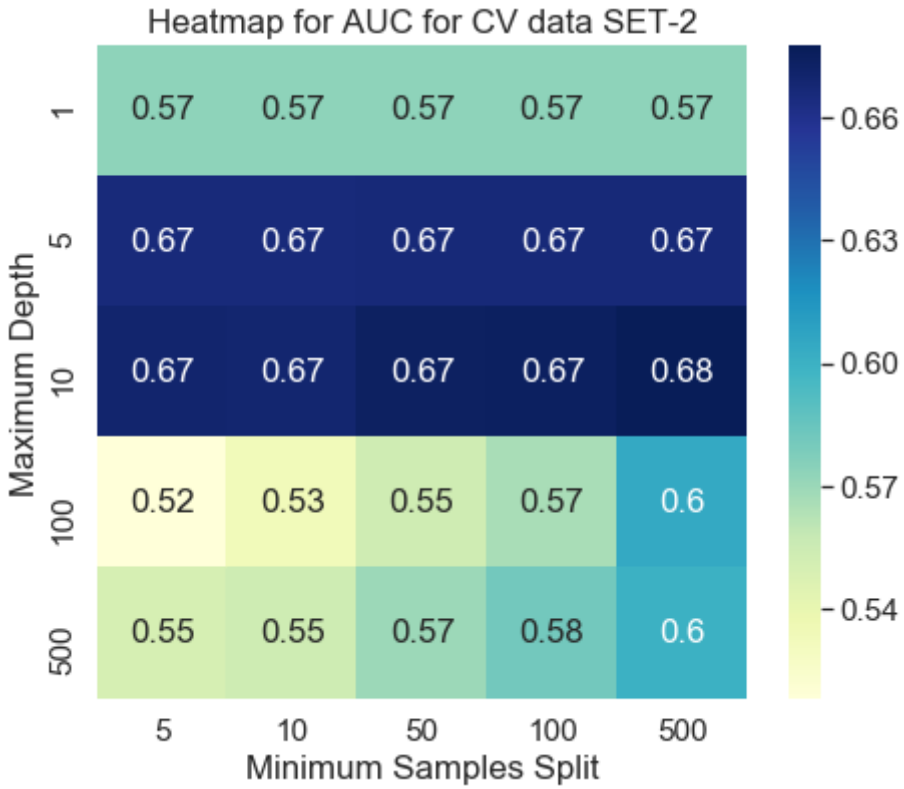
```
# adding the headings of every dimension
cv_auc_dataframe = cv_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum Sampl
es Split', values='CV_AUC_Score')
```

In [355]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(cv_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC for CV data SET-2')
```

Out[355]:

```
Text(0.5, 1.0, 'Heatmap for AUC for CV data SET-2')
```



Now seeing the two heatmaps from above the best value of max_depth is 10 and min_sample_split is 500

Training our final model based on the value of parameters received from above.

In [357]:

```
from sklearn.tree import DecisionTreeClassifier

model_decision_tree = DecisionTreeClassifier(max_depth = 10, min_samples_split = 500)

model_decision_tree.fit(X_train_merge_set_2,y_train)
```

Out[357]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=10,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=500,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best')
```

In [358]:

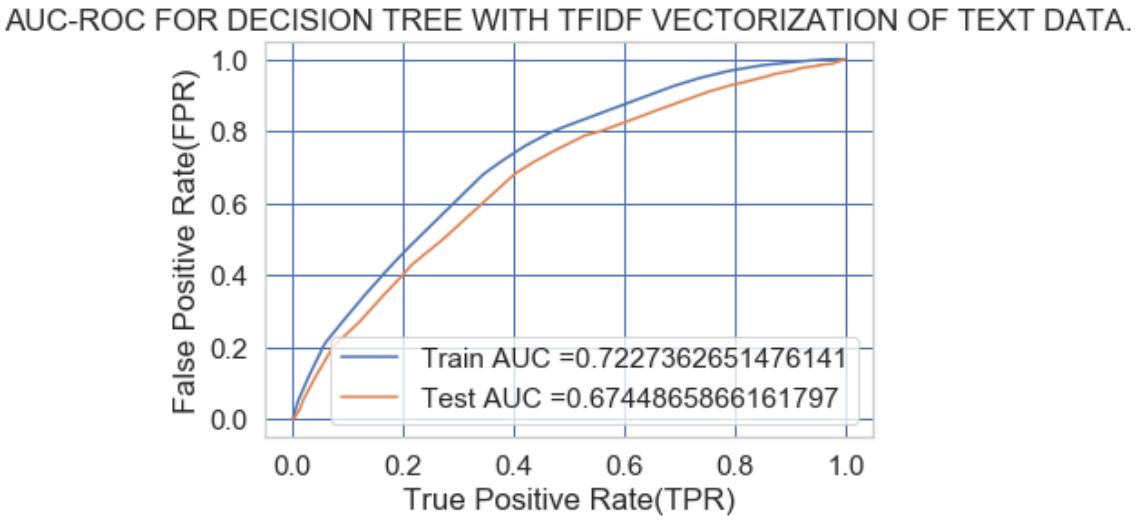
```
y_train_pred =  model_decision_tree.predict_proba(X_train_merge_set_2)[:,1]

y_test_pred =  model_decision_tree.predict_proba(X_test_merge_set_2)[:,1]
```

In [359]:

```
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [360]:

```
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.grid(b=True, which='major', color='b', linestyle='-')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC-ROC FOR DECISION TREE WITH TFIDF VECTORIZATION OF TEXT DATA.")

plt.show()
```



We received the train accuracy of 0.72 and test accuracy of 0.68 which is not that bad actually.

Plotting the Decision tree for set 2 using best parameters using graphviz

In [361]:

```
# refer -- > https://chrisalbon.com/machine_learning/trees_and_forests/visualize_a_decision_tree/
# refer ---> https://www.youtube.com/watch?v=XxJKt9s0Dlg

dot_data = tree.export_graphviz(model_decision_tree, out_file=None,feature_names=set_2_feature_names)

graph = graphviz.Source(dot_data)
graph.format = 'png'
graph.render("Set -2 Tree with TFIDF features with best parameters values found using gridsearch",view = True)
#graph = pydotplus.graph_from_dot_data(dot_data)

#Image(graph.create_png())
```

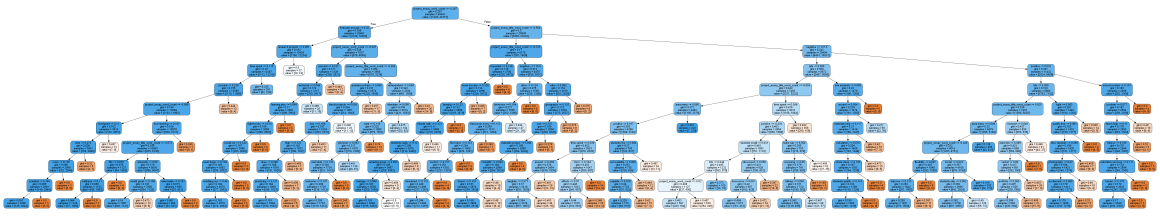
Out[361]:

```
'Set -2 Tree with TFIDF features with best parameters values found using g
ridsearch.png'
```

In [362]:

```
dot_data = tree.export_graphviz(model_decision_tree, out_file=None,feature_names=set_2_feature_names,filled=True,rounded=True)

graph = pydotplus.graph_from_dot_data(dot_data)

Image(graph.create_png())
```

Out[362]:



I tried many things but could not make it visible clearly due to large number of depth but I have received its image in my desktop which can be zoomed easily and seen.

Confusion matrix for above data

In [363]:

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

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

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

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

Confusion matrix for train and test data for TFIDF vectorization

In [364]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24991859713025238 for threshold 0.781
[[ 3780  3646]
 [ 7834 33781]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24954456577686931 for threshold 0.82
[[ 2613  2846]
 [ 6674 23919]]
```

Visually plotting the confusion matrix for training data

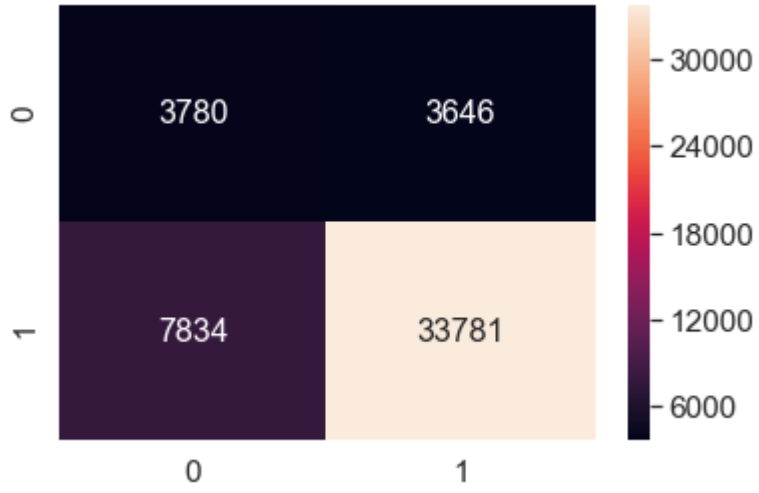
In [365]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt


df_cm = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred, tr_thresholds, trai
n_fpr, train_fpr)), range(2),
                     range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24991859713025238 for threshold 0.781
```

Out[365]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc24588b38>
```



Visually plotting the confusion matrix for test data

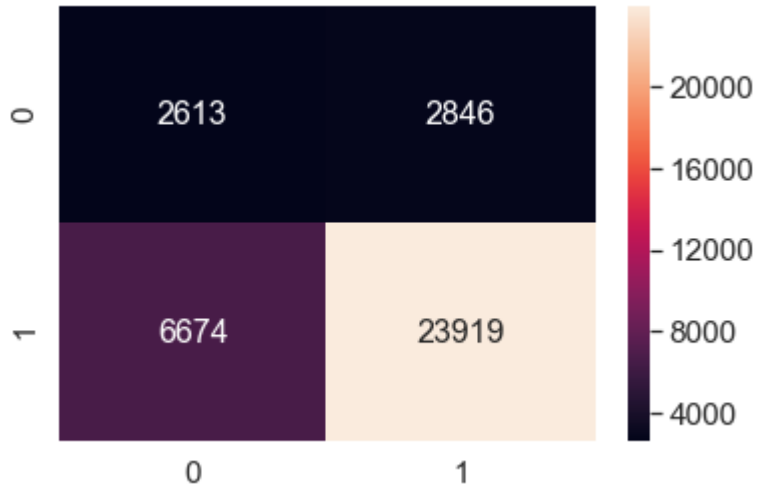
In [366]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_test = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred, tr_thresholds, t
est_fpr, test_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_test, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24954456577686931 for threshold 0.82
```

Out[366]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc2c391f98>
```



Analysis of False Positive Test Points On Our TFIDF VECTORIZED Essay Text.

In [367]:

```
text_tfidf_test.shape
```

Out[367]:

```
(36052, 5000)
```

In [368]:

```
type(essay_bow_test)
```

Out[368]:

```
scipy.sparse.csr.csr_matrix
```

Now out of all the 36052 test daTa points for TFIDF VECTORIZED essay we need to find out those which were declared as False Positives.

False Positive -->> Label which was predicted as Positive, but is actually Negative.

In [369]:

```
X_test_merge_set_2.shape
```

Out[369]:

```
(36052, 8108)
```

In [372]:

```
# getting all the words for which we have performed the vectorization
tfidf_words_list[0:20]
```

Out[372]:

```
['offering',
 'resources home',
 'use classroom',
 'new exciting',
 'copies',
 'socioeconomic backgrounds',
 'economy',
 'voices',
 'professional',
 'fluency comprehension',
 'would make',
 'art projects',
 'able participate',
 'classroom safe',
 'pe',
 'school work',
 'income households',
 'teachers',
 'work school',
 'list']
```

In [373]:

```
len(tfidf_words_list)
```

Out[373]:

```
5000
```

Now we have the predicted values in the y_test_pred variable

In [374]:

```
type(y_test_pred)
```

Out[374]:

```
numpy.ndarray
```

In [375]:

```
y_test_pred.shape
```

Out[375]:

```
(36052,)
```

In [376]:

```
y_test_pred[0:20]
```

Out[376]:

```
array([0.90590551, 0.94654605, 0.84762425, 0.59296482, 0.90327169,
       0.89911308, 0.76116259, 0.71902269, 0.76270733, 0.96540881,
       0.90590551, 0.91340782, 0.84762425, 0.89381197, 0.89381197,
       0.76116259, 0.91092328, 0.2       , 0.89381197, 0.89381197])
```

Now the above values are the predicted probability values of the data points whether they will belong to the positive class or the negative class.In the confusion matrix code starting part we can see the variable t which gives us the threshold value which means the value above which a data point is considered in the positive class.

For the test data it can be directly seen from confusion matrix code that threshold value = 0.82

We want all those points which have predict_proba values greater than 0.82 because they are the predicted positive points.

Along with them we will need data points whose actual value(y_test) = 0 because they will be the points whose actual values will be negative.¶

And after combining the above two conditions we will reach to the points whic were predicted to be positive but are actually negative which is actually the False positive points.

In [377]:

```
type(y_test)
```

Out[377]:

```
pandas.core.series.Series
```

In [378]:

```
y_test_list = list(y_test)
```

In [379]:

```
len(y_test_list)
```

Out[379]:

```
36052
```

In [380]:

```
threshold_value_test = 0.82
```

We will try to get the index of all those points which are false positives

In [394]:

```
fp_index = []
number_of_fp = 0
k = len(y_test_list)
for j in tqdm(range(k)):
    if y_test_list[j] == 0 and y_test_pred[j] >= threshold_value_test:
        fp_index.append(j)
        number_of_fp = number_of_fp + 1
```

```
100%|██████████████████████████████████████████████████████████████████████
██| 36052/36052 [00:00<00:00, 2780725.06it/s]
```

In [395]:

```
#fp_index[2450:2646]
```

In [396]:

```
len(fp_index)
```

Out[396]:

```
2608
```

So there are 2608 false positive points and yes they are different from the False positive shown in confusion matrix and this happened because the model was trained again for essay text and there is just a very small difference of value so no worries

Now I want the words at above indexes only

In [397]:

```
final_fp_words = []
for k in fp_index:
    if k<5000:
        final_fp_words.append(str(tfidf_words_list[k]))
```

In [398]:

```
len(final_fp_words)
```

Out[398]:

```
354
```

In [399]:

```
final_fp_words[0:20]
```

Out[399]:

```
['art projects',
 'live poverty',
 'new students',
 'due lack',
 'differentiation',
 'rate',
 'younger',
 'kiddos',
 'seniors',
 'classroom place',
 'celebrate',
 'morning meeting',
 'virtual',
 'classroom would',
 'hands projects',
 'novels',
 'classrooms',
 '2nd',
 'readers nannan',
 'around us']
```

In [400]:

```
final_fp_words == tfidf_words_list
```

Out[400]:

```
False
```

Creating Word Cloud for the False Positive Words From TFIDF VECTORIZED Essay

In [401]:

```
!pip install wordcloud
```

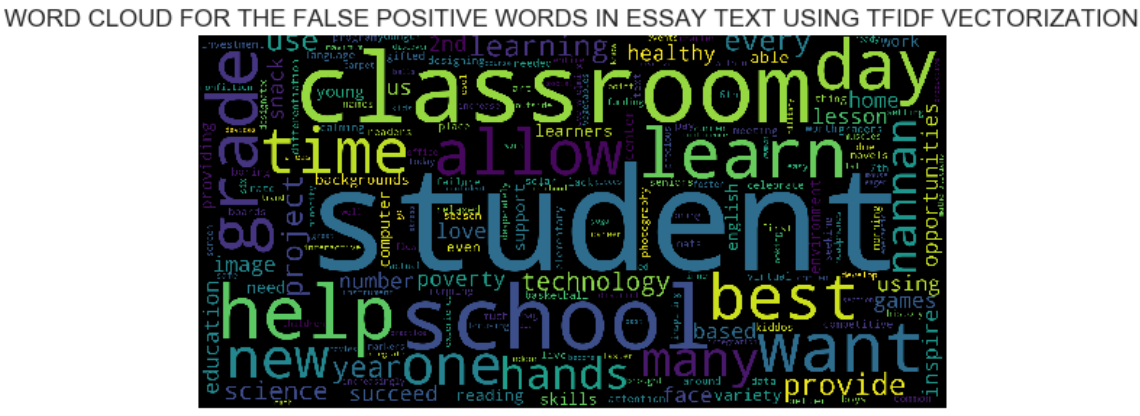
```
Requirement already satisfied: wordcloud in c:\users\rashu tyagi\anaconda3
\lib\site-packages (1.5.0)
Requirement already satisfied: numpy>=1.6.1 in c:\users\rashu tyagi\anacon
da3\lib\site-packages (from wordcloud) (1.16.2)
Requirement already satisfied: pillow in c:\users\rashu tyagi\anaconda3\li
b\site-packages (from wordcloud) (5.4.1)
```

In [402]:

```
# refer ---> https://www.geeksforgeeks.org/generating-word-cloud-python/

# importing all necessery modules
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt

stopwords = set(STOPWORDS)

#NOTE THAT WE NEED A SINGLE STRING WITH ALL THE WORDS SEPARATED BY SPACE WHOSE WORD CLO
UD WEWANT TO CREATE.

final_word_string = ' '
for words in final_fp_words:
    final_word_string = final_word_string + words + " "



wordcloud = WordCloud(width = 2000, height = 1000,background_color ='black',stopwords =
stopwords,min_font_size = 10).generate(final_word_string)
```

In [403]:

```
# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.title("WORD CLOUD FOR THE FALSE POSITIVE WORDS IN ESSAY TEXT USING TFIDF VECTORIZAT
ION")

plt.show()
```



Box-Plot for these False Positive Data Points.

For drawing the box plot we will need a dataframe with all these false positive data points

In [404]:

```
box_plot_dataframe_price = pd.DataFrame(X_test['price'])
```

In [405]:

```
# keeping only those indices which are refered as the false positves and removing all o
ther
box_plot_dataframe_price = box_plot_dataframe_price.iloc[fp_index,:]
```

In [406]:

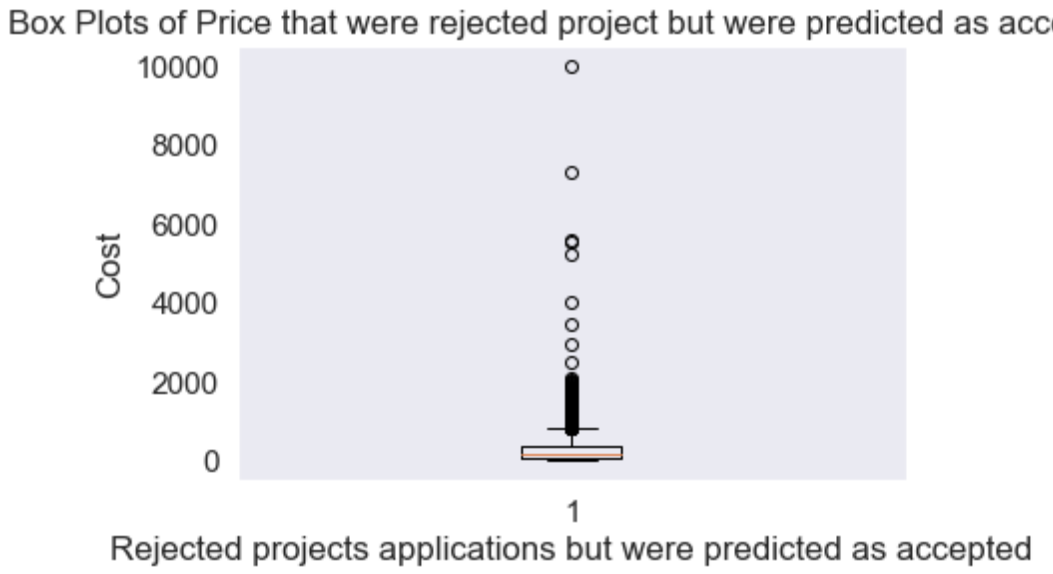
```
box_plot_dataframe_price.shape
```

Out[406]:

```
(2608, 1)
```

In [407]:

```
# creating the box plot

plt.boxplot(box_plot_dataframe_price.values)
plt.title('Box Plots of Price that were rejected project but were predicted as accepte
d')
plt.xlabel('Rejected projects applications but were predicted as accepted')
plt.ylabel('Cost')
plt.grid()
plt.show()
```



From above we can see that most of the wrongly accepted projects are costing very less at around 500 they might be accepted wrongly due to less cost included but actually they were rejected

PDF with the Teacher_number_of_previously_posted_projects of the False Positive data points.

In [408]:

```
pdf_dataframe_teacher_number_of_previously_posted_projects = pd.DataFrame(X_test['teacher_number_of_previously_posted_projects'])
```

In [409]:

```
# keeping only those indices which are refered as the false positves and removing all other
pdf_dataframe_teacher_number_of_previously_posted_projects = pdf_dataframe_teacher_number_of_previously_posted_projects.iloc[fp_index,:]
```

In [410]:

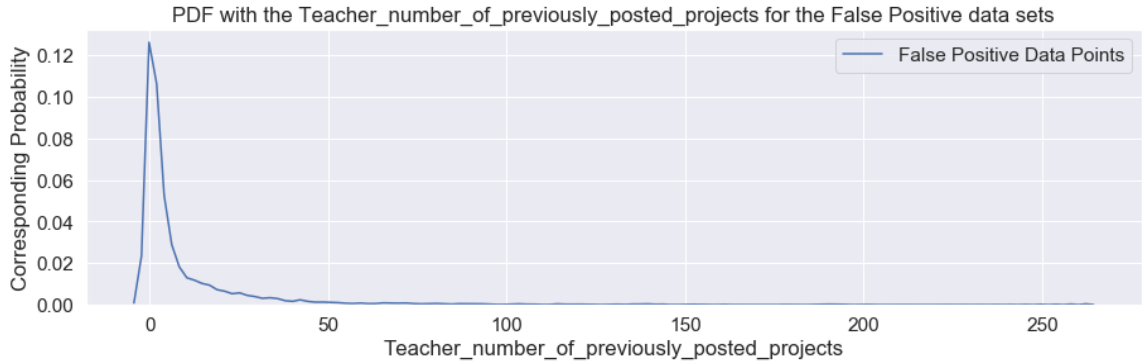
```
pdf_dataframe_teacher_number_of_previously_posted_projects.shape
```

Out[410]:

```
(2608, 1)
```

In [411]:

```
# plotting the pdf

plt.figure(figsize=(15,4))
sns.distplot(pdf_dataframe_teacher_number_of_previously_posted_projects.values, hist=False, label="False Positive Data Points")
plt.title('PDF with the Teacher_number_of_previously_posted_projects for the False Positive data sets')
plt.xlabel('Teacher_number_of_previously_posted_projects')
plt.ylabel('Corresponding Probability')
plt.legend()
plt.show()
```



From above we can say that most of the projects which were predicted as accpeted but were actually rejected have number of previously posted projects by a teacher in the range of 0-15 only.¶

In [ ]:

```
```

Set 3 : categorical, numerical features + project_title(AVG W2V) + preprocessed_essay (AVG W2V)

Now we need to merge all the numerical vectors(categorical features,text features,numerical features) given above for set-3 which we created using different methods

In [412]:

```
from scipy.sparse import hstack

X_train_merge_set_3 = hstack((categories_one_hot_train,subcategories_one_hot_train,teacher_prefix_one_hot_train,project_grade_categories_one_hot_train,school_state_categories_one_hot_train,avg_w2v_vectors_train,avg_w2v_vectors_project_title_train,price_train,quantity_train,prev_projects_train,project_title_word_count_train,project_essay_word_count_train,neutral_train,positive_train,negative_train,compound_train)).tocsr()
X_test_merge_set_3 = hstack((categories_one_hot_test,subcategories_one_hot_test,teacher_prefix_one_hot_test,project_grade_categories_one_hot_test,school_state_categories_one_hot_test,avg_w2v_vectors_test,avg_w2v_vectors_project_title_test,price_test,quantity_test,prev_projects_test,project_title_word_count_test,project_essay_word_count_test,neutral_test,positive_test,negative_test,compound_test)).tocsr()
X_cv_merge_set_3 = hstack((categories_one_hot_cv,subcategories_one_hot_cv,teacher_prefix_one_hot_cv,project_grade_categories_one_hot_cv,school_state_categories_one_hot_cv,avg_w2v_vectors_cv,avg_w2v_vectors_project_title_cv,price_cv,quantity_cv,prev_projects_cv,project_title_word_count_cv,project_essay_word_count_cv,neutral_cv,positive_cv,negative_cv,compound_cv)).tocsr()
```

In [413]:

```
# this will be our finally created data matrix dimensions

print(X_train_merge_set_3.shape, y_train.shape)
print(X_test_merge_set_3.shape, y_test.shape)
print(X_cv_merge_set_3.shape, y_cv.shape)
```

```
(49041, 708) (49041,)
(36052, 708) (36052,)
(24155, 708) (24155,)
```

HERE WE WILL NOT BE USING GRAPHVIZ BECAUSE WORD2VEC MODEL DOESNOT RETURN US THE ACTUAL WORDS AND HENCE WE CANNOT VISUALIZE THE WORDS USING GRAPHVIZ

Using GridSearchCV for the purpose of hyperparameter tuning and finding the best parameters for our final model.

Applying the GridsearchCV to find the best value of hyperparameters

In [416]:

```
decision_tree = DecisionTreeClassifier()

tuned_parameters = {'max_depth':[1,5,10,100,500], 'min_samples_split': [5,10,50,100,500
]}
```

In [417]:

```
# refer --> https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
# cv = 10 means 10 fold crossvalidation

clf = GridSearchCV(decision_tree,tuned_parameters,cv=10,scoring = 'roc_auc',n_jobs=-1,verbose=10)

clf.fit(X_train_merge_set_3,y_train)
```

```
Fitting 10 folds for each of 25 candidates, totalling 250 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 6 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    7.9s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    9.7s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:   20.0s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:   25.9s
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:   38.0s
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:   45.3s
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:   58.6s
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:  1.7min
[Parallel(n_jobs=-1)]: Done  73 tasks      | elapsed:  2.6min
[Parallel(n_jobs=-1)]: Done  86 tasks      | elapsed:  3.4min
[Parallel(n_jobs=-1)]: Done 101 tasks      | elapsed:  4.9min
[Parallel(n_jobs=-1)]: Done 116 tasks      | elapsed:  7.6min
[Parallel(n_jobs=-1)]: Done 133 tasks      | elapsed: 10.1min
[Parallel(n_jobs=-1)]: Done 150 tasks      | elapsed: 12.8min
[Parallel(n_jobs=-1)]: Done 169 tasks      | elapsed: 38.1min
[Parallel(n_jobs=-1)]: Done 188 tasks      | elapsed: 55.3min
[Parallel(n_jobs=-1)]: Done 209 tasks      | elapsed: 73.8min
[Parallel(n_jobs=-1)]: Done 230 tasks      | elapsed: 91.2min
[Parallel(n_jobs=-1)]: Done 250 out of 250 | elapsed: 107.1min finished
```

Out[417]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'max_depth': [1, 5, 10, 100, 500], 'min_samples_split': [5, 10, 50, 100, 500]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=10)
```

In [418]:

```
#https://stackoverflow.com/questions/44947574/what-is-the-meaning-of-mean-test-score-in-cv-result

train_auc= clf.cv_results_['mean_train_score']
cv_auc = clf.cv_results_['mean_test_score']
```

Now we have 3 things to compare 1.) we need to compare our parameters max_depth and min_samples_split with our train_auc score on a single graph 2.) we need to compare our parameters max_depth and min_samples_split with our cv_auc score on a single graph

In order to have all 3 things on the same graph what i will be using here is the heatmaps representations.

Comparing for train_auc

In [419]:

```
# importing the required libraries

import numpy as np
import seaborn as sns
```

In [420]:

```
max_depth = [1,5,10,100,500]
min_samples_split = [5,10,50,100,500]
```

In [421]:

```
train_auc.shape
```

Out[421]:

```
(25,)
```

In [422]:

```
cv_auc.shape
```

Out[422]:

```
(25,)
```

In [425]:

```
# creating dataframe to plot heatmap for train_auc

#train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Sp
lit':min_samples_split, 'Train_AUC_Score':train_auc})

# using the above code I was getting error that all arrays must be of same size and thi
s happened because max_depth and min_samples_split are of size 5 each only
# while train_auc has size of 25

# now in order to deal with it I will keep the value of max_depth five times each so wi
th mij samples split because the algorithm of
# grid search cv works this way only that it compapres one with all elements of opther
 parameter list and so on.

# WE CAN DO IT VICE VERSA ALSO LIKE WRITING 1,5,10,100,500 5 TIMES FOR MAX DEPTH AND WR
ITING 55555,10,10,10,10,10 SO ON FOR MIN_SAMPLES_SPLIT
```

In [426]:

```
max_depth = [1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,100,100,100,100,100,500,500,500,500,500
]
min_samples_split = [5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,
10,50,100,500]
```

In [427]:

```
len(max_depth)
```

Out[427]:

25

In [428]:

```
len(min_samples_split)
```

Out[428]:

25

In [429]:

```
# creating dataframe to plot heatmap for train_auc

train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Spl
it':min_samples_split, 'Train_AUC_Score':train_auc})
```

In [430]:

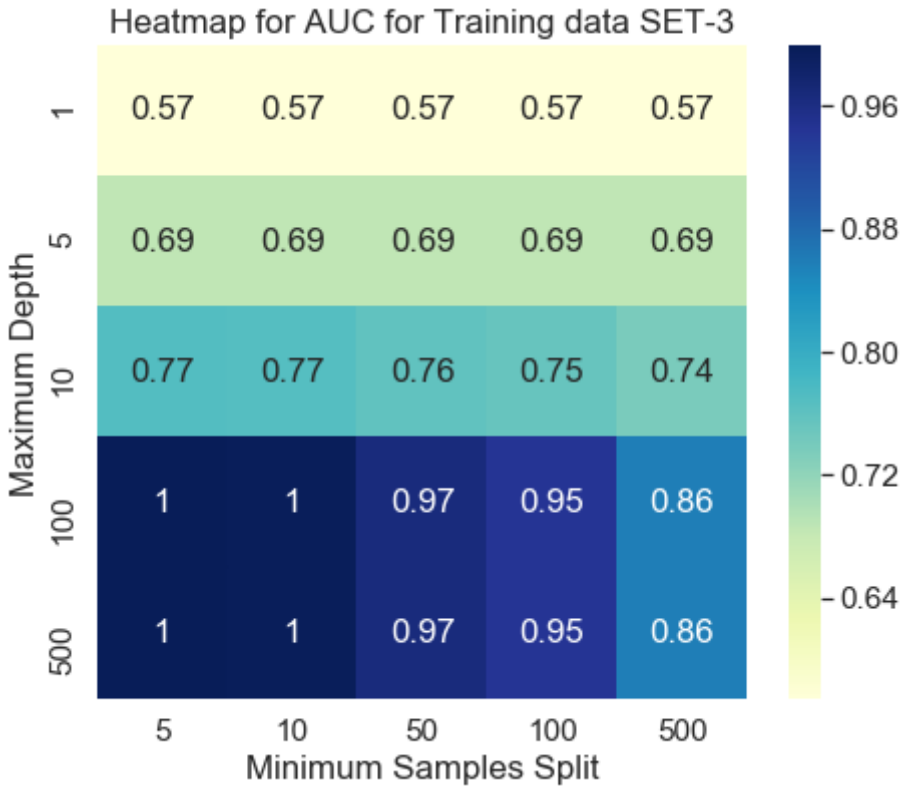
```
# adding the headings of every dimesion
train_auc_dataframe = train_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum
Samples Split', values='Train_AUC_Score')
```

In [431]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(train_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC
 for Training data SET-3')
```

Out[431]:

```
Text(0.5, 1.0, 'Heatmap for AUC for Training data SET-3')
```



Comparing for cv_auc

In [432]:

```
# creating dataframe to plot heatmap for cv_auc

cv_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Split'
:min_samples_split, 'CV_AUC_Score':cv_auc})
```

In [433]:

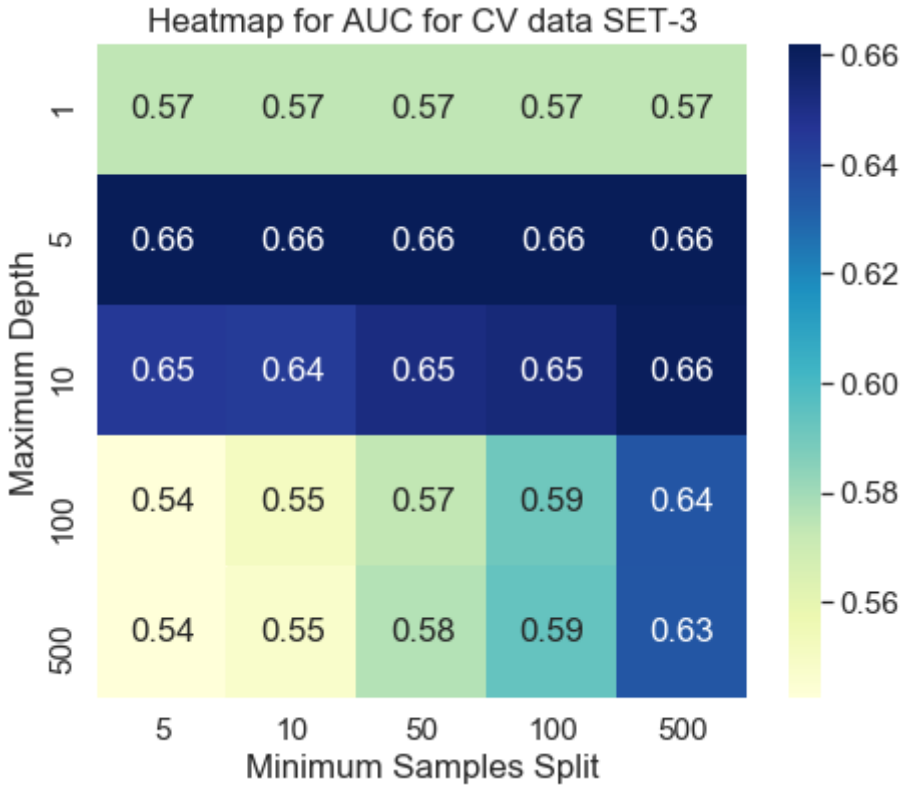
```
# adding the headings of every dimension
cv_auc_dataframe = cv_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum Sampl
es Split', values='CV_AUC_Score')
```

In [434]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(cv_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC for
CV data SET-3')
```

Out[434]:

```
Text(0.5, 1.0, 'Heatmap for AUC for CV data SET-3')
```



Now seeing the two heatmaps from above the best value of max_depth is 5 and min_sample_split is 50

Training our final model based on the value of parameters received from above

In [436]:

```
from sklearn.tree import DecisionTreeClassifier

model_decision_tree = DecisionTreeClassifier(max_depth = 5, min_samples_split = 50)

model_decision_tree.fit(X_train_merge_set_3,y_train)
```

Out[436]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=50,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best')
```

In [437]:

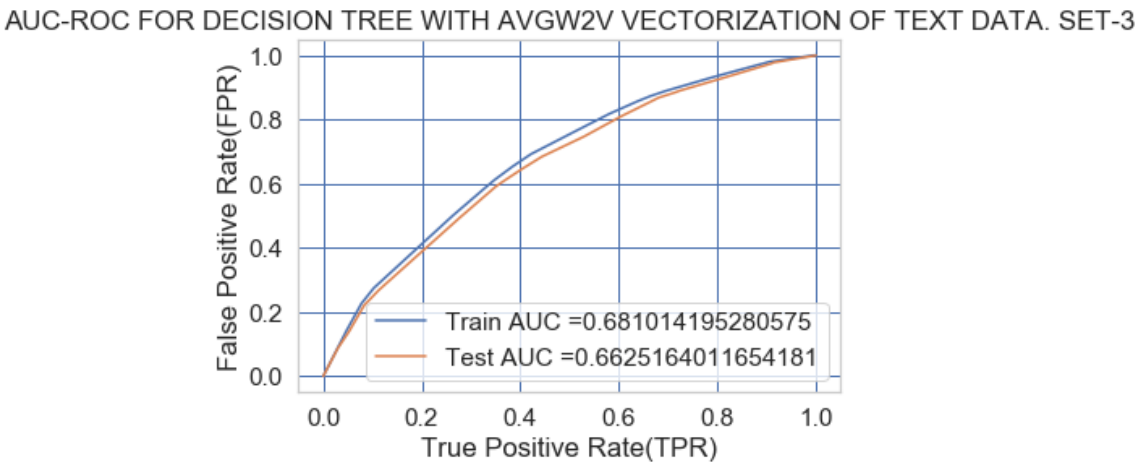
```
y_train_pred =  model_decision_tree.predict_proba(X_train_merge_set_3)[:,1]

y_test_pred =  model_decision_tree.predict_proba(X_test_merge_set_3)[:,1]
```

In [438]:

```
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [439]:

```
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.grid(b=True, which='major', color='b', linestyle='-')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC-ROC FOR DECISION TREE WITH AVGW2V VECTORIZATION OF TEXT DATA. SET-3")

plt.show()
```



We received the train accuracy of 0.68 and test accuracy of 0.66 which is not that bad actually.

Confusion matrix for above data

Confusion matrix for train and test data for BOW vectorization

In [440]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24998866634136951 for threshold 0.816
[[ 3688  3738]
 [10125 31490]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2491355818345225 for threshold 0.816
[[ 2569  2890]
 [ 7745 22848]]
```

Visually plotting the confusion matrix for training data

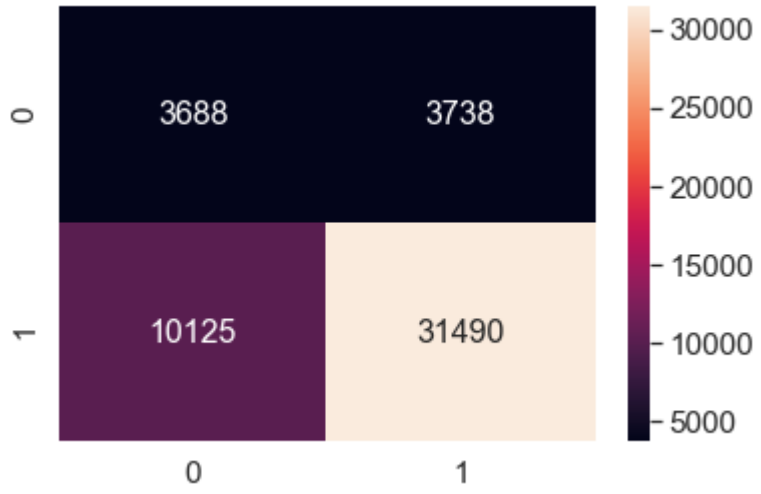
In [441]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt


df_cm = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred, tr_thresholds, trai
n_fpr, train_fpr)), range(2),
                     range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24998866634136951 for threshold 0.816
```

Out[441]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc3c315240>
```



Visually plotting the confusion matrix for test data

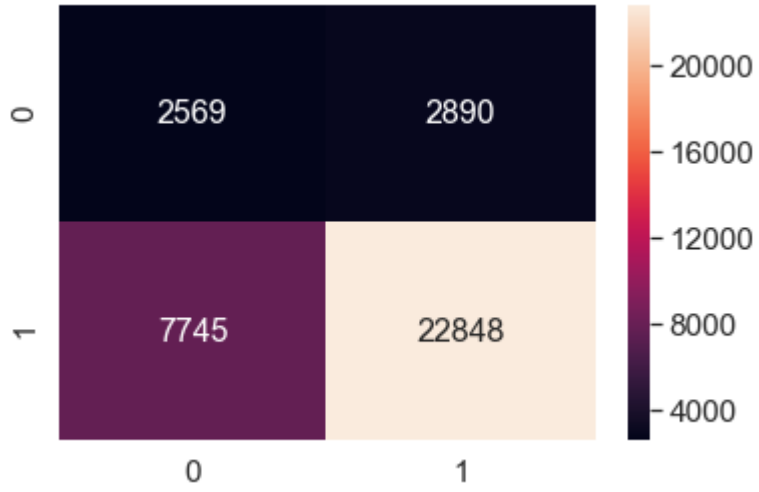
In [442]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_test = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred, tr_thresholds, t
est_fpr, test_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_test, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.2491355818345225 for threshold 0.816
```

Out[442]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc2ad276d8>
```



In [ ]:

```

```

Set 4 : Categorical, Numerical features + Project_title(TFIDF W2V) + Preprocessed_essay (TFIDF W2V)

Now we need to merge all the numerical vectors(categorical features,text features,numerical features) given above for set-3 which we created using different methods

In [445]:

```
from scipy.sparse import hstack

X_train_merge_set_4 = hstack((categories_one_hot_train,subcategories_one_hot_train,teac
her_prefix_one_hot_train,project_grade_categories_one_hot_train,school_state_categories
_one_hot_train,tfidf_w2v_vectors_text_train,tfidf_w2v_vectors_project_title_train,price
_train,quantity_train,prev_projects_train,project_title_word_count_train,project_essay_
word_count_train,neutral_train,positive_train,negative_train,compound_train)).tocsr()
X_test_merge_set_4 = hstack((categories_one_hot_test,subcategories_one_hot_test,teacher
_prefix_one_hot_test,project_grade_categories_one_hot_test,school_state_categories_one_
hot_test,tfidf_w2v_vectors_text_test,tfidf_w2v_vectors_project_title_test,price_test,qu
antity_test,prev_projects_test,project_title_word_count_test,project_essay_word_count_t
est,neutral_test,positive_test,negative_test,compound_test)).tocsr()
X_cv_merge_set_4 = hstack((categories_one_hot_cv,subcategories_one_hot_cv,teacher_prefi
x_one_hot_cv,project_grade_categories_one_hot_cv,school_state_categories_one_hot_cv,tfi
df_w2v_vectors_text_cv,tfidf_w2v_vectors_project_title_cv,price_cv,quantity_cv,prev_pro
jects_cv,project_title_word_count_cv,project_essay_word_count_cv,neutral_cv,positive_cv
,negative_cv,compound_cv)).tocsr()
```

In [446]:

```
# this will be our finally created data matrix dimensions

print(X_train_merge_set_4.shape, y_train.shape)
print(X_test_merge_set_4.shape, y_test.shape)
print(X_cv_merge_set_4.shape,y_cv.shape)
```

```
(49041, 708) (49041,)
(36052, 708) (36052,)
(24155, 708) (24155,)
```

HERE WE WILL NOT BE USING GRAPHVIZ BECAUSE WORD2VEC MODEL DOESNOT RETURN US THE ACTUAL WORDS AND HENCE WE CANNOT VISUALIZE THE WORDS USING GRAPHVIZ

Using GridSearchCV for the purpose of hyperparameter tuning and finding the best parameters for our final model.

Importing the required modules

In [452]:

```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_le
arning_lecture_2/Machine%20Learning%20Lecture%202.html

from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
```

Applying the GridsearchCV to find the best value of hyperparameters

In [453]:

```
decision_tree = DecisionTreeClassifier()

tuned_parameters = {'max_depth':[1,5,10,100,500], 'min_samples_split': [5,10,50,100,500
]}
```

In [454]:

```
# refer --> https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.G
ridSearchCV.html
# cv = 10 means 10 fold crossvalidation

clf = GridSearchCV(decision_tree,tuned_parameters,cv=10,scoring = 'roc_auc',n_jobs=-1,
verbose=10)

clf.fit(X_train_merge_set_4,y_train)
```

```
Fitting 10 folds for each of 25 candidates, totalling 250 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 6 concurrent worker
s.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    8.8s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:   14.3s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:   25.1s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:   30.6s
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:   41.8s
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:   52.1s
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:  1.1min
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done  73 tasks      | elapsed:  3.0min
[Parallel(n_jobs=-1)]: Done  86 tasks      | elapsed:  3.8min
[Parallel(n_jobs=-1)]: Done 101 tasks      | elapsed:  5.4min
[Parallel(n_jobs=-1)]: Done 116 tasks      | elapsed:  8.0min
[Parallel(n_jobs=-1)]: Done 133 tasks      | elapsed: 11.4min
[Parallel(n_jobs=-1)]: Done 150 tasks      | elapsed: 14.8min
[Parallel(n_jobs=-1)]: Done 169 tasks      | elapsed: 32.7min
[Parallel(n_jobs=-1)]: Done 188 tasks      | elapsed: 46.9min
[Parallel(n_jobs=-1)]: Done 209 tasks      | elapsed: 61.7min
[Parallel(n_jobs=-1)]: Done 230 tasks      | elapsed: 77.6min
[Parallel(n_jobs=-1)]: Done 250 out of 250 | elapsed: 90.1min finished
```

Out[454]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight=None, criterion='gin
i', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'max_depth': [1, 5, 10, 100, 500], 'min_samples_split':
[5, 10, 50, 100, 500]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=10)
```

In [455]:

```
#https://stackoverflow.com/questions/44947574/what-is-the-meaning-of-mean-test-score-in-cv-result

train_auc= clf.cv_results_['mean_train_score']
cv_auc = clf.cv_results_['mean_test_score']
```

Now we have 3 things to compare 1.) we need to compare our parameters max_depth and min_samples_split with our train_auc score on a single graph 2.) we need to compare our parameters max_depth and min_samples_split with our cv_auc score on a single graph¶

In order to have all 3 things on the same graph what i will be using here is the heatmaps representations.

Comparing for train_auc

In [456]:

```
# importing the required libraries

import numpy as np
import seaborn as sns
```

In [457]:

```
max_depth = [1,5,10,100,500]
min_samples_split = [5,10,50,100,500]
```

In [458]:

```
train_auc.shape
```

Out[458]:

```
(25,)
```

In [459]:

```
cv_auc.shape
```

Out[459]:

```
(25,)
```

In [460]:

```
# creating dataframe to plot heatmap for train_auc

#train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Split':min_samples_split, 'Train_AUC_Score':train_auc})

# using the above code I was getting error that all arrays must be of same size and this happened because max_depth and min_samples_split are of size 5 each only
# while train_auc has size of 25

# now in order to deal with it I will keep the value of max_depth five times each so with mij samples split because the algorithm of
# grid search cv works this way only that it compapres one with all elements of opther parameter list and so on.
```

In [461]:

```
max_depth = [1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,100,100,100,100,100,500,500,500,500,500]
min_samples_split = [5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500]
```

In [462]:

```
len(max_depth)
```

Out[462]:

```
25
```

In [463]:

```
len(min_samples_split)
```

Out[463]:

```
25
```

In [464]:

```
# creating dataframe to plot heatmap for train_auc

train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Split':min_samples_split, 'Train_AUC_Score':train_auc})
```

In [465]:

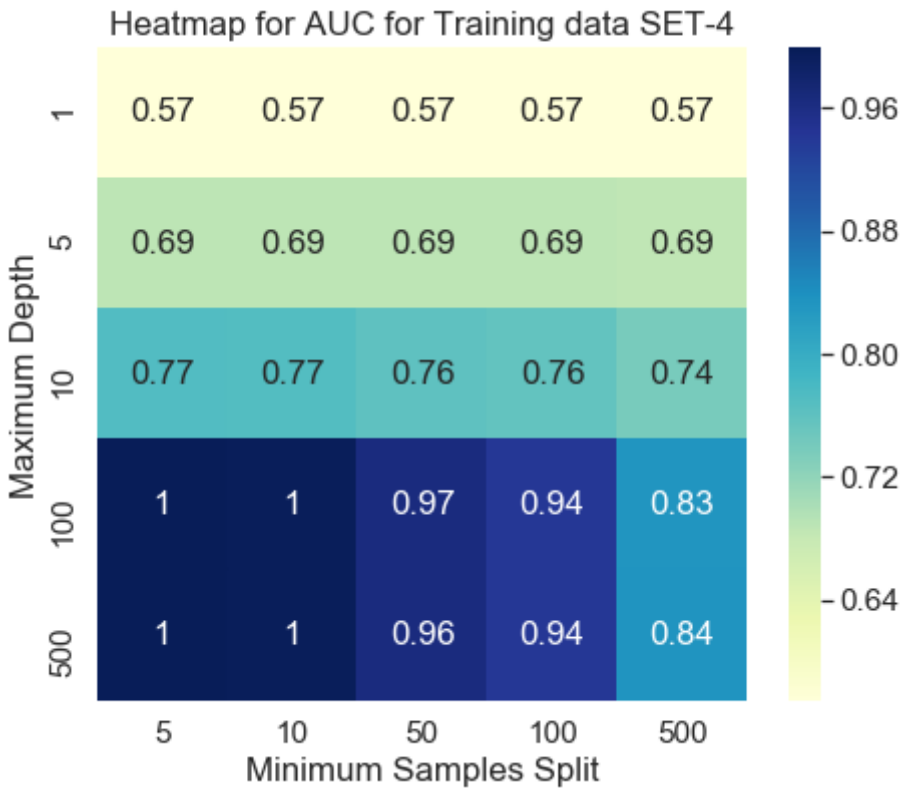
```
# adding the headings of every dimesion
train_auc_dataframe = train_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum Samples Split', values='Train_AUC_Score')
```

In [466]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(train_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC
 for Training data SET-4')
```

Out[466]:

```
Text(0.5, 1.0, 'Heatmap for AUC for Training data SET-4')
```



Comparing for cv_auc

In [467]:

```
# creating dataframe to plot heatmap for cv_auc

cv_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Split'
:min_samples_split, 'CV_AUC_Score':cv_auc})
```

In [468]:

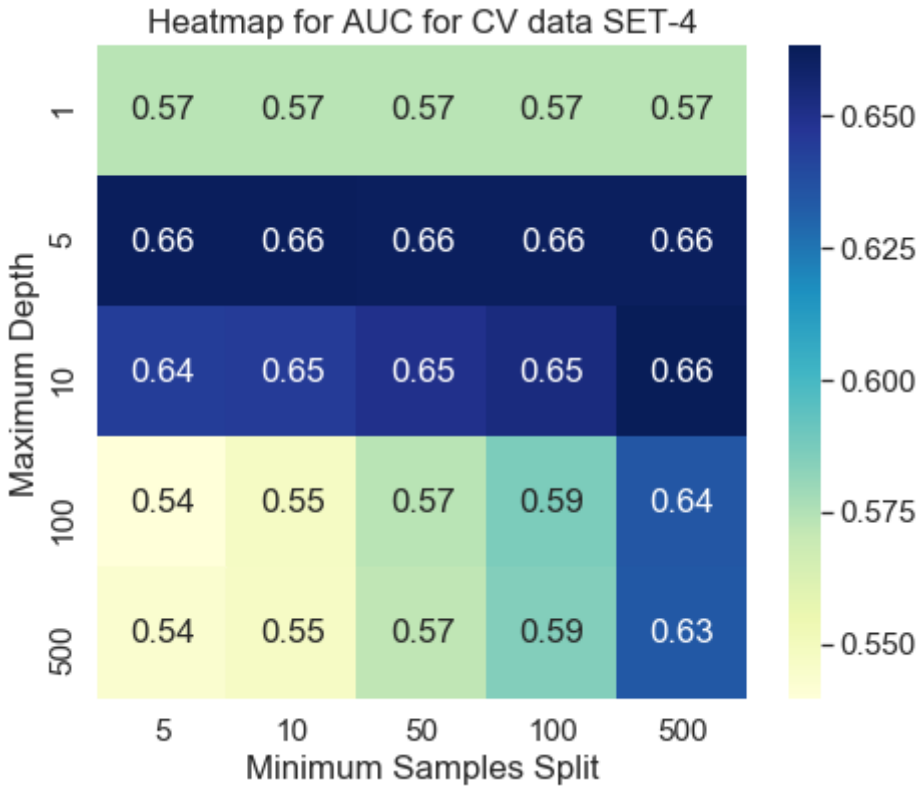
```
# adding the headings of every dimension
cv_auc_dataframe = cv_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum Sampl
es Split', values='CV_AUC_Score')
```

In [469]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(cv_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC for
CV data SET-4')
```

Out[469]:

```
Text(0.5, 1.0, 'Heatmap for AUC for CV data SET-4')
```



Now seeing the two heatmaps from above the best value of max_depth is 5 and min_sample_split is 100

Training our final model based on the value of parameters received from above

In [470]:

```
from sklearn.tree import DecisionTreeClassifier

model_decision_tree = DecisionTreeClassifier(max_depth = 5, min_samples_split = 100)

model_decision_tree.fit(X_train_merge_set_4,y_train)
```

Out[470]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=100,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best')
```

In [471]:

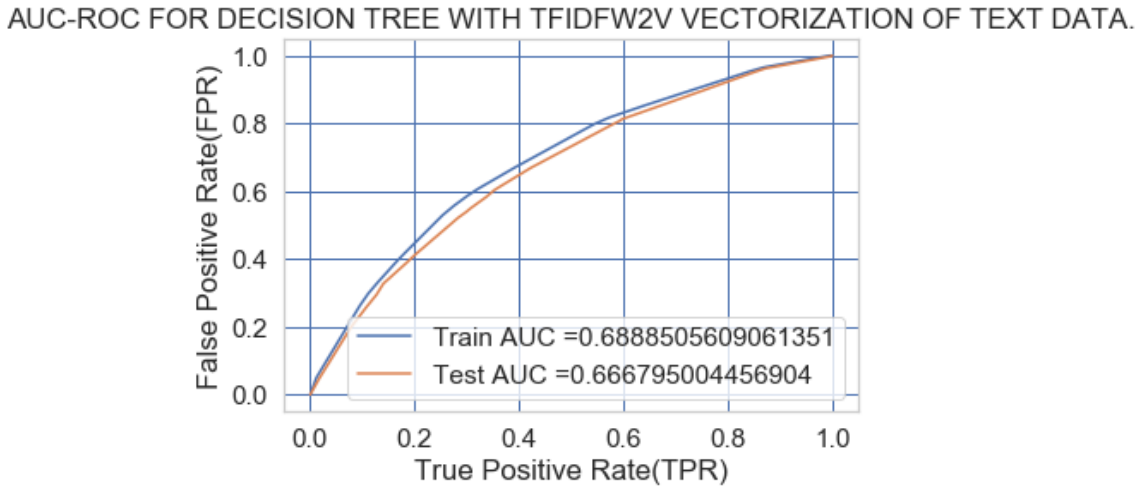
```
y_train_pred =  model_decision_tree.predict_proba(X_train_merge_set_4)[:,1]

y_test_pred =  model_decision_tree.predict_proba(X_test_merge_set_4)[:,1]
```

In [472]:

```
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [473]:

```
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.grid(b=True, which='major', color='b', linestyle='-')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC-ROC FOR DECISION TREE WITH TFIDFW2V VECTORIZATION OF TEXT DATA.")

plt.show()
```



We received the train accuracy of 0.69 and test accuracy of 0.67 which is not that bad actually

Confusion matrix for above data

In [474]:

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

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

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

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

Confusion matrix for train and test data for TFIDFW2V vectorization

In [475]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24802522332022303 for threshold 0.825
[[ 3383  4043]
 [ 8406 33209]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24450951305441646 for threshold 0.825
[[ 2325  3134]
 [ 6348 24245]]
```

Visually plotting the confusion matrix for training data

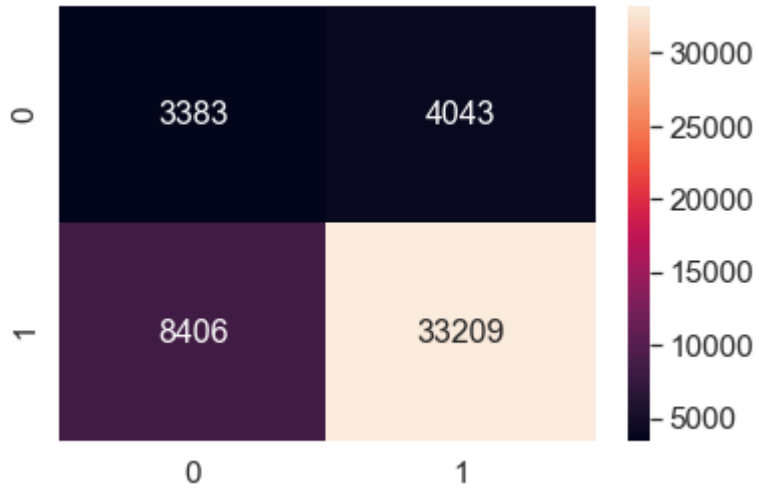
In [476]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt


df_cm = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred, tr_thresholds, trai
n_fpr, train_fpr)), range(2),
                     range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24802522332022303 for threshold 0.825
```

Out[476]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc390611d0>
```



Visually plotting the confusion matrix for test data

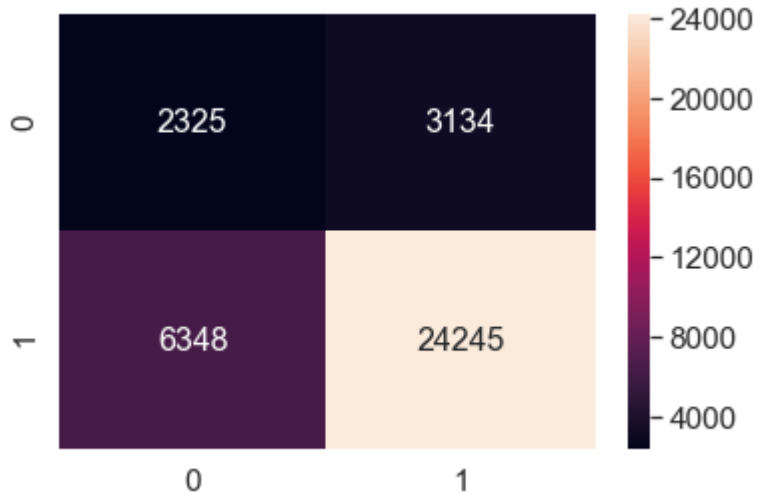
In [477]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_test = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred, tr_thresholds, t
est_fpr, test_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_test, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24450951305441646 for threshold 0.825
```

Out[477]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc2c39f4a8>
```



NOW LET US PERFORM ONE MORE INTERESTING TASK.

Machine learning is all about the features of our data set so let us try to select the best 5000 features from our SET-2 Data and train the decision tre based on those features and let us see how does accuracy improves on doing that.

Best features can be selected using the following function in sklearn

In [479]:

```
# refere this to get the best features -- > https://scikit-learn.org/stable/modules/gen
erated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.fea
ture_importances_
```

We must keep in mind that The importance of a feature is computed as the (normalized) total reduction of the criterion brought by that feature. It is also known as the Gini importance.

SET-2 Data

In [157]:

```
from scipy.sparse import hstack

X_train_merge_set_2 = hstack((categories_one_hot_train,subcategories_one_hot_train,teac
her_prefix_one_hot_train,project_grade_categories_one_hot_train,school_state_categories
_one_hot_train,text_tfidf_train,project_title_tfidf_train,price_train,quantity_train,pr
ev_projects_train,project_title_word_count_train,project_essay_word_count_train,neutral
_train,positive_train,negative_train,compound_train)).tocsr()
X_test_merge_set_2 = hstack((categories_one_hot_test,subcategories_one_hot_test,teacher
_prefix_one_hot_test,project_grade_categories_one_hot_test,school_state_categories_one_
hot_test,text_tfidf_test,project_title_tfidf_test,price_test,quantity_test,prev_project
s_test,project_title_word_count_test,project_essay_word_count_test,neutral_test,positiv
e_test,negative_test,compound_test)).tocsr()
X_cv_merge_set_2 = hstack((categories_one_hot_cv,subcategories_one_hot_cv,teacher_prefi
x_one_hot_cv,project_grade_categories_one_hot_cv,school_state_categories_one_hot_cv,tex
t_tfidf_cv,title_tfidf_cv,price_cv,quantity_cv,prev_projects_cv,project_title_word_coun
t_cv,project_essay_word_count_cv,neutral_cv,positive_cv,negative_cv,compound_cv)).tocsr
()
```

In [158]:

```
# this will be our finally created data matrix dimensions

print(X_train_merge_set_2.shape, y_train.shape)
print(X_test_merge_set_2.shape, y_test.shape)
print(X_cv_merge_set_2.shape, y_cv.shape)
```

```
(49041, 8108) (49041,)
(36052, 8108) (36052,)
(24155, 8108) (24155,)
```

Let us just train our Decision tree model based on above data.

In [159]:

```
# refer --> https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTree
Classifier.html

from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier(max_depth = 8)
```

In [160]:

```
clf = decision_tree.fit(X_train_merge_set_2,y_train)
```

Now from the above trained Decision tree classifier we want the best features which hold the maximum importance.

In [161]:

```
# refer ---> https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier.feature_importances_
# refer --> https://stackoverflow.com/questions/49170296/scikit-learn-feature-importance-calculation-in-decision-trees

important_features_from_DT = decision_tree.feature_importances_
```

In [162]:

```
type(important_features_from_DT)
```

Out[162]:

```
numpy.ndarray
```

In [163]:

```
len(important_features_from_DT)
```

Out[163]:

```
8108
```

In [165]:

```
#important_features_from_DT[0:1000]
```

From above we can see thatmost of the features have 0 importance hence they should not be present in our final training set.

In [166]:

```
important_features_from_DT_list = list(important_features_from_DT)
```

Now I am planning to get the index of all those features which have importance greater than 0 and top 5000 of them so firstly let us remove all those featurees which have 0 importance.

In [167]:

```
len(important_features_from_DT_list)
```

Out[167]:

```
8108
```

In [168]:

```
final_list_important_features = []
```

In [169]:

```
for k in range(len(important_features_from_DT_list)):
    if important_features_from_DT_list[k] != 0:
        final_list_important_features.append(k)
```

In [170]:

```
len(final_list_important_features)
```

Out[170]:

```
110
```

So we have 110 features only which hold some importance for the model.

Here is the list of all those indexes which represent the important features.

In [171]:

```
print(final_list_important_features)
```

```
[27, 36, 113, 141, 209, 239, 310, 422, 476, 614, 657, 698, 775, 786, 848,
861, 1115, 1119, 1140, 1285, 1311, 1418, 1423, 1465, 1510, 1515, 1543, 156
0, 1565, 1600, 1624, 1670, 1715, 1798, 1854, 1903, 1921, 1978, 2074, 2105,
2275, 2305, 2340, 2405, 2448, 2455, 2467, 2478, 2506, 2667, 2671, 2753, 28
12, 2816, 2860, 2873, 2923, 2924, 2931, 2979, 2996, 3010, 3013, 3088, 317
7, 3198, 3231, 3288, 3335, 3439, 3477, 3518, 3555, 3608, 4117, 4232, 4357,
4375, 4378, 4388, 4421, 4511, 4562, 4569, 4614, 4644, 4695, 4699, 4716, 47
26, 4765, 4794, 4798, 4808, 4816, 4887, 4983, 5060, 5097, 5136, 5466, 616
5, 6538, 6702, 7230, 8099, 8100, 8101, 8103, 8106]
```

Selecting only those features from our set -2 training set whose indexes are present in the "final_list_important_features" list

In [ ]:

```
# I was getting error that iloc was not found because i used iloc on something which is
not even a dataframe firstly
```

In [183]:

```
type(X_train_merge_set_2)
```

Out[183]:

```
scipy.sparse.csr.csr_matrix
```

In [182]:

```
# so now converting back to dense matrix and then converting into the dataframe
k = pd.DataFrame(X_train_merge_set_2.todense())
```

In [180]:

```
X_train_final_set_2_important_features = k.iloc[:,final_list_important_features]
```

In [181]:

```
X_train_final_set_2_important_features.shape
```

Out[181]:

```
(49041, 110)
```

Now X_train_final_set_2_important_features is our final dataset of set 2 containing all the important features whic have feature_importance greater than 0

We will need the test set also so copying the same steps to the test dataset also.

In [184]:

```
type(X_test_merge_set_2)
```

Out[184]:

```
scipy.sparse.csr.csr_matrix
```

In [188]:

```
X_test_merge_set_2.shape
```

Out[188]:

```
(36052, 8108)
```

In [185]:

```
# so now converting back to dense matrix and then converting into the dataframe
j = pd.DataFrame(X_test_merge_set_2.todense())
```

In [189]:

```
j.shape
```

Out[189]:

```
(36052, 8108)
```

In [190]:

```
X_test_final_set_2_important_features = j.iloc[:,final_list_important_features]
```

In [191]:

```
X_test_final_set_2_important_features.shape
```

Out[191]:

```
(36052, 110)
```

We can do the same for cross validation sewt also but we will be using the Grid Search CV with 10 fold crossvalidation hence no need to have the crosvalidation set here.

Applying Decision Tree on the dataframe obtained from above using the best features.

In [241]:

```
# refer --> https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTree
Classifier.html

from sklearn.tree import DecisionTreeClassifier
decision_tree = DecisionTreeClassifier(max_depth = 3)
```

In [242]:

```
clf = decision_tree.fit(X_train_final_set_2_important_features,y_train)
```

Now the next important task for us is to visualize this decision tree for that we will need to use the library graphviz

Graphviz requires all the feature names hence we need to extract all the feature names first.

We have already done with Extracting feature names from essay on set-2 for graphviz

Now we will keep only those which fits our important featueres indexes.

In [243]:

```
len(set_2_feature_names)
```

Out[243]:

```
8108
```

In [244]:

```
final_important_set_2_feature_names = []

for i in range(len(set_2_feature_names)):
    if i in final_list_important_features:
        final_important_set_2_feature_names.append(set_2_feature_names[i])
```

In [245]:

```
len(final_important_set_2_feature_names)
```

Out[245]:

```
110
```

So these are our important features

In [246]:

```
#final_important_set_2_feature_names
```

Using graphviz to visualize the decision tree

In [247]:

```
!pip install pydot
```

```
Requirement already satisfied: pydot in c:\users\rashu tyagi\anaconda3\lib
\site-packages (1.4.1)
Requirement already satisfied: pyparsing>=2.1.4 in c:\users\rashu tyagi\an
aconda3\lib\site-packages (from pydot) (2.3.1)
```

In [248]:

```
from IPython.display import Image
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
import pydot
import pydotplus
from sklearn import tree
import graphviz
```

In [249]:

```
!pip install pydotplus
```

```
Requirement already satisfied: pydotplus in c:\users\rashu tyagi\anaconda3
\lib\site-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\rashu tyagi\an
aconda3\lib\site-packages (from pydotplus) (2.3.1)
```

In [250]:

```
!pip install graphviz
```

```
Requirement already satisfied: graphviz in c:\users\rashu tyagi\anaconda3
\lib\site-packages (0.11.1)
```

In [251]:

```
# refer -- > https://chrisalbon.com/machine_learning/trees_and_forests/visualize_a_deci
sion_tree/
# refer ---> https://www.youtube.com/watch?v=XxJKt9s0Dlg

dot_data = tree.export_graphviz(decision_tree, out_file=None,feature_names=final_import
ant_set_2_feature_names)

graph = graphviz.Source(dot_data)
graph.format = 'png'
graph.render("Set -2 Tree with TFIDF features With Important features selected",view =
True)
#graph = pydotplus.graph_from_dot_data(dot_data)

#Image(graph.create_png())
```

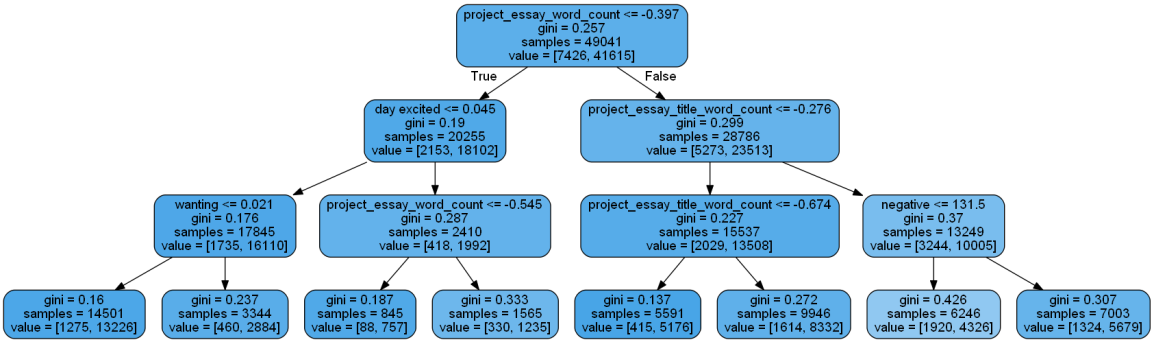
Out[251]:

```
'Set -2 Tree with TFIDF features With Important features selected.png'
```

In [252]:

```
dot_data = tree.export_graphviz(decision_tree, out_file=None,feature_names=final_import
ant_set_2_feature_names,filled=True,rounded=True)

graph = pydotplus.graph_from_dot_data(dot_data)

Image(graph.create_png())
```

Out[252]:



Now the Important task which comes is to know what best hyperparameters should we pass to the decision tree classifier to get the best results in terms of accuracy.

For the task of getting the best hyperparameters we will be doing hyperparameter tuning and as we know that we have two hyperparameters in case of decision tree and they are 1.) Maximum depth of the decision tree and 2.) minimum number of sample split

Minimum number of sample split means what should be the minimum number of data points present in a node in order to split the node.

Importing the required modules

In [253]:

```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Machine%20Learning%20Lecture%202.html

from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
```

Applying the GridsearchCV to find the best value of hyperparameters

In [254]:

```
decision_tree = DecisionTreeClassifier()

tuned_parameters = {'max_depth':[1,5,10,100,500], 'min_samples_split': [5,10,50,100,500]}
```

In [256]:

```
# refer --> https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
# cv = 10 means 10 fold crossvalidation

clf = GridSearchCV(decision_tree,tuned_parameters,cv=10,scoring = 'roc_auc',n_jobs=-1,verbose=10)

clf.fit(X_train_final_set_2_important_features,y_train)
```

```
Fitting 10 folds for each of 25 candidates, totalling 250 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 6 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:    1.5s
[Parallel(n_jobs=-1)]: Done   6 tasks      | elapsed:    1.6s
[Parallel(n_jobs=-1)]: Done  13 tasks      | elapsed:    2.1s
[Parallel(n_jobs=-1)]: Done  20 tasks      | elapsed:    2.5s
[Parallel(n_jobs=-1)]: Done  29 tasks      | elapsed:    3.0s
[Parallel(n_jobs=-1)]: Done  38 tasks      | elapsed:    3.4s
[Parallel(n_jobs=-1)]: Done  49 tasks      | elapsed:    3.9s
[Parallel(n_jobs=-1)]: Done  60 tasks      | elapsed:    5.5s
[Parallel(n_jobs=-1)]: Done  73 tasks      | elapsed:    7.1s
[Parallel(n_jobs=-1)]: Done  86 tasks      | elapsed:    8.8s
[Parallel(n_jobs=-1)]: Done 101 tasks      | elapsed:   11.3s
[Parallel(n_jobs=-1)]: Done 116 tasks      | elapsed:   14.0s
[Parallel(n_jobs=-1)]: Done 133 tasks      | elapsed:   17.6s
[Parallel(n_jobs=-1)]: Done 150 tasks      | elapsed:   20.8s
[Parallel(n_jobs=-1)]: Done 169 tasks      | elapsed:   30.2s
[Parallel(n_jobs=-1)]: Done 188 tasks      | elapsed:   38.5s
[Parallel(n_jobs=-1)]: Done 209 tasks      | elapsed:   47.6s
[Parallel(n_jobs=-1)]: Done 230 tasks      | elapsed:   57.7s
[Parallel(n_jobs=-1)]: Done 250 out of 250 | elapsed:  1.1min finished
```

Out[256]:

```
GridSearchCV(cv=10, error_score='raise-deprecating',
       estimator=DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False, random_state=None,
            splitter='best'),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'max_depth': [1, 5, 10, 100, 500], 'min_samples_split': [5, 10, 50, 100, 500]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=10)
```

In [257]:

```
#https://stackoverflow.com/questions/44947574/what-is-the-meaning-of-mean-test-score-in-cv-result

train_auc= clf.cv_results_['mean_train_score']
cv_auc = clf.cv_results_['mean_test_score']
```

Now we have 3 things to compare 1.) we need to compare our parameters max_depth and min_samples_split with our train_auc score on a single graph 2.) we need to compare our parameters max_depth and min_samples_split with our cv_auc score on a single graph

In order to have all 3 things on the same graph what i will be using here is the heatmaps representations.

Comparing for train_auc

In [258]:

```
# importing the required libraries

import numpy as np
import seaborn as sns
```

In [259]:

```
max_depth = [1,5,10,100,500]
min_samples_split = [5,10,50,100,500]
```

In [260]:

```
train_auc.shape
```

Out[260]:

```
(25,)
```

In [261]:

```
cv_auc.shape
```

Out[261]:

```
(25,)
```

In [262]:

```
# creating dataframe to plot heatmap for train_auc

#train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Split':min_samples_split, 'Train_AUC_Score':train_auc})

# using the above code I was getting error that all arrays must be of same size and this happened because max_depth and min_samples_split are of size 5 each only
# while train_auc has size of 25

# now in order to deal with it I will keep the value of max_depth five times each so with mij samples split because the algorithm of
# grid search cv works this way only that it compapres one with all elements of opther parameter list and so on.


# WE CAN DO IT VICE VERSA ALSO LIKE WRITING 1,5,10,100,500 5 TIMES FOR MAX DEPTH AND WRITING 55555,10,10,10,10,10 SO ON FOR MIN_SAMPLES_SPLIT
```

In [263]:

```
max_depth = [1,1,1,1,1,5,5,5,5,5,10,10,10,10,10,100,100,100,100,100,500,500,500,500,500]
min_samples_split = [5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500,5,10,50,100,500]
```

In [264]:

```
len(max_depth)
```

Out[264]:

25

In [265]:

```
len(min_samples_split)
```

Out[265]:

25

In [266]:

```
# creating dataframe to plot heatmap for train_auc

train_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Split':min_samples_split, 'Train_AUC_Score':train_auc})
```

In [267]:

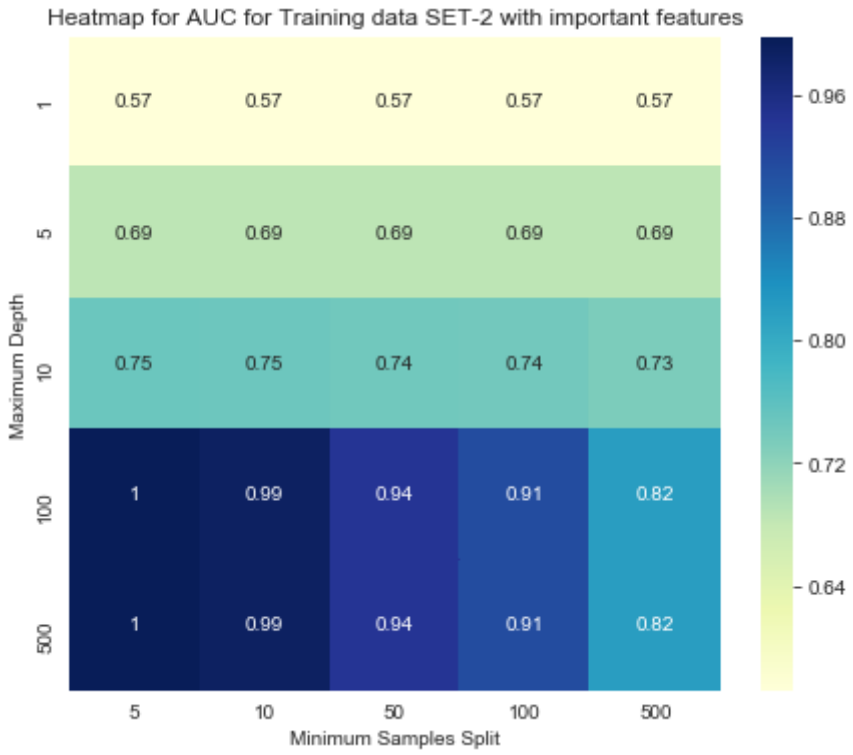
```
# adding the headings of every dimesion
train_auc_dataframe = train_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum Samples Split', values='Train_AUC_Score')
```

In [268]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(train_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC
 for Training data SET-2 with important features')
```

Out[268]:

```
Text(0.5, 1.0, 'Heatmap for AUC for Training data SET-2 with important fea
tures')
```



Comparing for cv_auc

In [269]:

```
# creating dataframe to plot heatmap for cv_auc

cv_auc_dataframe =pd.DataFrame(data={'Maximum Depth':max_depth, 'Minimum Samples Split'
:min_samples_split, 'CV_AUC_Score':cv_auc})
```

In [270]:

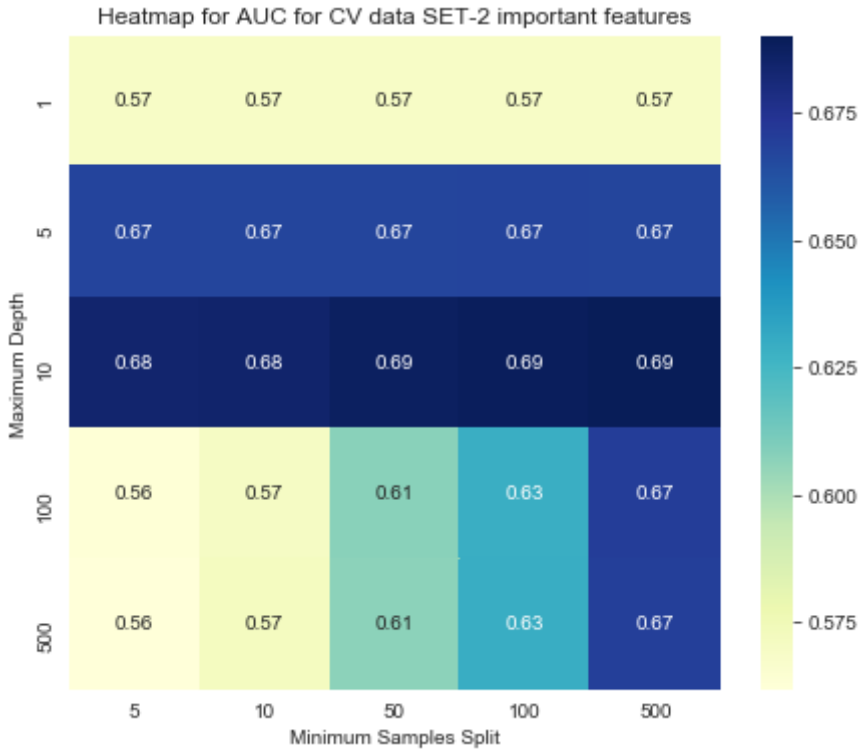
```
# adding the headings of every dimension
cv_auc_dataframe = cv_auc_dataframe.pivot(index='Maximum Depth', columns='Minimum Sampl
es Split', values='CV_AUC_Score')
```

In [271]:

```
sns.set_style("whitegrid")
plt.figure(figsize=(16,6))
plt.subplot(1,2,1)
sns.heatmap(cv_auc_dataframe, annot=True, cmap="YlGnBu").set_title('Heatmap for AUC for
CV data SET-2 important features')
```

Out[271]:

```
Text(0.5, 1.0, 'Heatmap for AUC for CV data SET-2 important features')
```



Now seeing the two heatmaps from above the best value of max_depth is 5 and min_sample_split is 100

Training our final model based on the value of parameters received from above.

In [272]:

```
from sklearn.tree import DecisionTreeClassifier

model_decision_tree = DecisionTreeClassifier(max_depth = 10, min_samples_split = 100)

model_decision_tree.fit(X_train_final_set_2_important_features,y_train)
```

Out[272]:

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=10,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=100,
            min_weight_fraction_leaf=0.0, presort=False, random_state=Non
e,
            splitter='best')
```

In [273]:

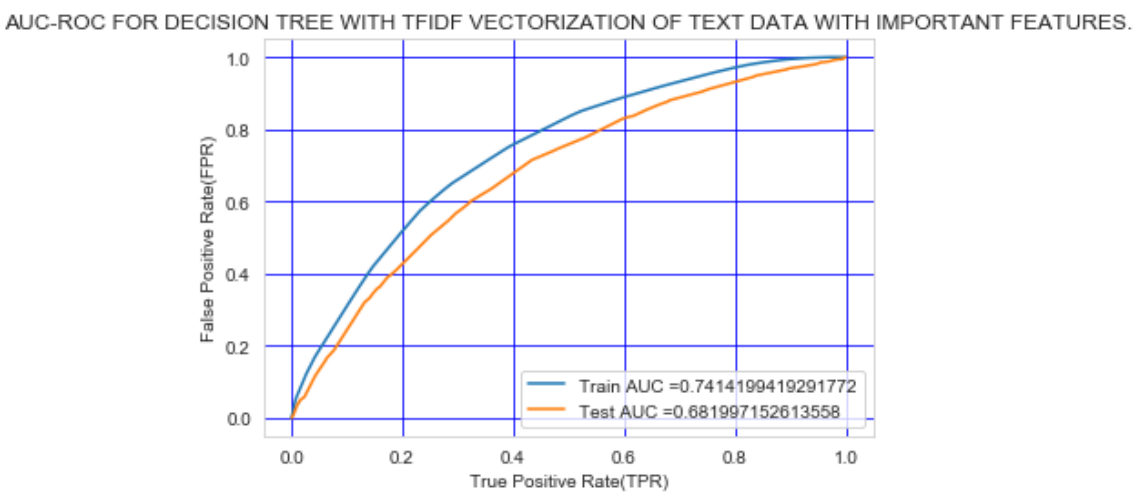
```
y_train_pred =  model_decision_tree.predict_proba(X_train_final_set_2_important_feature
s)[:,1]

y_test_pred =  model_decision_tree.predict_proba(X_test_final_set_2_important_features)
[:,1]
```

In [274]:

```
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [275]:

```
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.grid(b=True, which='major', color='b', linestyle='-')
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC-ROC FOR DECISION TREE WITH TFIDF VECTORIZATION OF TEXT DATA WITH IMPORTA
NT FEATURES.")

plt.show()
```



We received the train accuracy of 0.74 and test accuracy of 0.69 which is not that bad actually

Plotting the Decision tree for set 2 with best features using best parameters using graphviz

In [276]:

```
# refer -- > https://chrisalbon.com/machine_learning/trees_and_forests/visualize_a_decision_tree/
# refer ---> https://www.youtube.com/watch?v=XxJKt9s0Dlg

dot_data = tree.export_graphviz(model_decision_tree, out_file=None,feature_names=final_important_set_2_feature_names)

graph = graphviz.Source(dot_data)
graph.format = 'png'
graph.render("Set -2 Tree with TFIDF features with best features with best parameters values found using gridsearch",view = True)
#graph = pydotplus.graph_from_dot_data(dot_data)

#Image(graph.create_png())
```

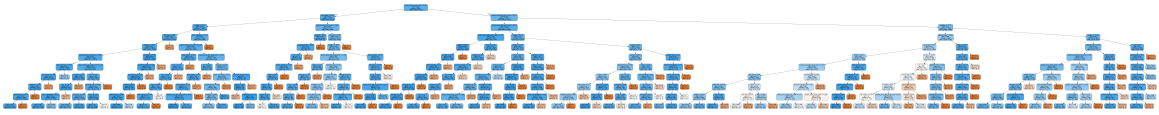
Out[276]:

```
'Set -2 Tree with TFIDF features with best features with best parameters v
alues found using gridsearch.png'
```

In [277]:

```
dot_data = tree.export_graphviz(model_decision_tree, out_file=None,feature_names=final_important_set_2_feature_names,filled=True,rounded=True)

graph = pydotplus.graph_from_dot_data(dot_data)

Image(graph.create_png())
```

Out[277]:



I tried many things but could not make it visible clearly due to large number of depth but I have received its image in my desktop which can be zoomed easily and seen.

Confusion matrix for above data

In [278]:

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

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

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

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

Confusion matrix for train and test data for TFIDF vectorization

In [279]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_f
pr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr
)))
```

```
====================================================================================
==========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24998367953157208 for threshold 0.812
[[ 3683  3743]
 [ 6744 34871]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2496810056119187 for threshold 0.821
[[ 2827  2632]
 [ 7734 22859]]
```

Visually plotting the confusion matrix for training data

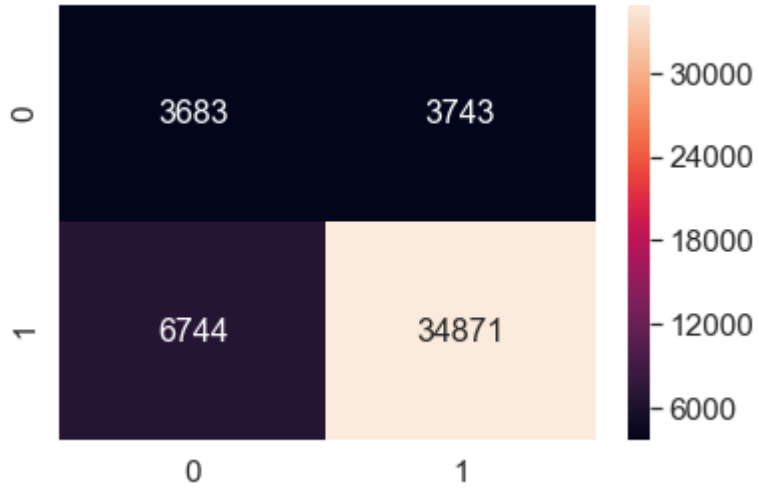
In [280]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt


df_cm = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred, tr_thresholds, trai
n_fpr, train_fpr)), range(2),
                     range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.24998367953157208 for threshold 0.812
```

Out[280]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x213554de0b8>
```



Visually plotting the confusion matrix for test data

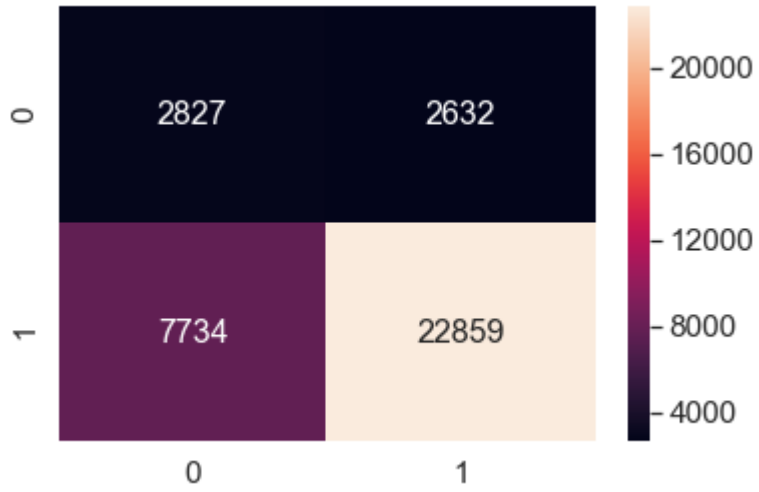
In [281]:

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

import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt

df_cm_test = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred, tr_thresholds, t
est_fpr, test_fpr)), range(2),
                  range(2))
#plt.figure(figsize = (10,7))
sn.set(font_scale=1.4)#for label size
sn.heatmap(df_cm_test, annot=True,annot_kws={"size": 16},fmt='g')# font size
```

```
the maximum value of tpr*(1-fpr) 0.2496810056119187 for threshold 0.821
```

Out[281]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2134a39c9e8>
```



Summary

In [282]:

```
from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prett
ytable
x = PrettyTable()
x.field_names = ["Vectorizer Used", "Model For Training", "Hyperparameters taken(min sa
mples split,max_depth)", "Train AUC Score"
, "Test AUC Score"]
x.add_row(["BOW", "Decision Trees","(100,10)", 0.73, 0.69])
x.add_row(["TFIDF", "Decision Trees", "(500,10)", 0.72, 0.68])
x.add_row(["AVG W2V", "Decision Trees", "(50,5)", 0.68, 0.66])
x.add_row(["TFIDF W2V", "Decision Trees", "(100,5)", 0.69, 0.67])
x.add_row(["TFIDF- Best Features", "Decision Trees", "(100,5)", 0.74, 0.69])
```

In [283]:

```
print(x)
```

```
+----------------------+--------------------+-----------------------------
-----------------------+-----------------+----------------+
|   Vectorizer Used    | Model For Training | Hyperparameters taken(min sa
mples split,max_depth) | Train AUC Score | Test AUC Score |
+----------------------+--------------------+-----------------------------
-----------------------+-----------------+----------------+
|         BOW          |   Decision Trees   |                      (100,1
0)                     |       0.73      |      0.69      |
|        TFIDF         |   Decision Trees   |                      (500,1
0)                     |       0.72      |      0.68      |
|       AVG W2V        |   Decision Trees   |                       (50,5)
|       0.68      |      0.66      |
|      TFIDF W2V       |   Decision Trees   |                      (100,5)
|       0.69      |      0.67      |
| TFIDF- Best Features |   Decision Trees   |                      (100,5)
|       0.74      |      0.69      |
+----------------------+--------------------+-----------------------------
-----------------------+-----------------+----------------+
```

In [ ]:

```

```