DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located ([Two-letter U.S. postal code (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!</code`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:

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

In [3]:

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

```
Number of data points in train data (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]:

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

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

Out[4]:

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

1.2 preprocessing of `project_subject_categories`

In [5]:

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

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

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [6]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

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

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

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

1.3 Text preprocessing

Removing null values from project essay 3 & 4

In [7]:
```
# check if we have any nan values are there in the column
print(project_data['project_essay_3'].isnull().values.any())
print("number of nan values",project_data['project_essay_3'].isnull().values.sum())
```

```
True
number of nan values 105490
```

In [8]:
```
#Replacing the Nan values with most frequent value in the column
project_data['project_essay_3']=project_data['project_essay_3'].fillna(' ')
```

In [9]:
```
# check if we have any nan values are there in the column
print(project_data['project_essay_3'].isnull().values.any())
print("number of nan values",project_data['project_essay_3'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [10]:
```
# check if we have any nan values are there in the column
print(project_data['project_essay_4'].isnull().values.any())
print("number of nan values",project_data['project_essay_4'].isnull().values.sum())
```

```
True
number of nan values 105490
```

In [11]:
```
#Replacing the Nan values with most frequent value in the column
project_data['project_essay_4']=project_data['project_essay_4'].fillna(' ')
```

In [12]:
```
# check if we have any nan values are there in the column
print(project_data['project_essay_4'].isnull().values.any())
print("number of nan values",project_data['project_essay_4'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [13]:

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

In [14]:

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

Out[14]:

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

In [15]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [16]:
```python
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. We are
a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our sc
hool. \r\n\r\n We have over 24 languages represented in our English Learner program with students at eve
ry level of mastery.  We also have over 40 countries represented with the families within our school.  E
ach student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, belie
fs, and respect.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein  Our E
nglish learner's have a strong support system at home that begs for more resources.  Many times our pare
nts are learning to read and speak English along side of their children.  Sometimes this creates barrier
s for parents to be able to help their child learn phonetics, letter recognition, and other reading skil
ls.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the Engl
ish language even if no one at home is able to assist.  All families with students within the Level 1 pr
oficiency status, will be a offered to be a part of this program.  These educational videos will be spec
ially chosen by the English Learner Teacher and will be sent home regularly to watch.  The videos are to
help the child develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will
have the opportunity to check out a dvd player to use for the year.  The plan is to use these videos and
educational dvd's for the years to come for other EL students.\r\n
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at least m
ost of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 st
udents, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get together a
nd celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes that st
udents wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and gam
es. At the end of the year the school hosts a carnival to celebrate the hard work put in during the scho
ol year, with a dunk tank being the most popular activity.My students will use these five brightly color
ed Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in
the classroom and not enough for each student to have an individual one, they will be used in a variety
of ways. During independent reading time they will be used as special chairs students will each use on o
ccasion. I will utilize them in place of chairs at my small group tables during math and reading times.
The rest of the day they will be used by the students who need the highest amount of movement in their l
ife in order to stay focused on school.\r\n\r\nWhenever asked what the classroom is missing, my students
always say more Hokki Stools. They can't get their fill of the 5 stools we already have. When the studen
ts are sitting in group with me on the Hokki Stools, they are always moving, but at the same time doing
their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. There are always students who head over to the kidney table to get one of the stools who are disapp
ointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The H
okki stools will be a compromise that allow my students to do desk work and move at the same time. These
stools will help students to meet their 60 minutes a day of movement by allowing them to activate their
core muscles for balance while they sit. For many of my students, these chairs will take away the barrie
r that exists in schools for a child who can't sit still.
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desk
s, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to cr
eate a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nMy class is
made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I sch
ool, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our schoo
l is an \"open classroom\" concept, which is very unique as there are no walls separating the classroom
s. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the in
formation and experiences and keep on wanting more.With these resources such as the comfy red throw pill
ows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the m
ood in our classroom setting to be one of a themed nautical environment. Creating a classroom environmen
t is very important in the success in each and every child's education. The nautical photo props will be
used with each child as they step foot into our classroom for the first time on Meet the Teacher evenin
g. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready
for their first day of 4th grade.  This kind gesture will set the tone before even the first day of scho
ol! The nautical thank you cards will be used throughout the year by the students as they create thank y
ou cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a
fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on
resources to get our classroom ready. Please consider helping with this project to make our new school y
ear a very successful one. Thank you!
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive del
ays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest
```

```
working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I
teach in a Title I school where most of the students receive free or reduced price lunch.  Despite their
disabilities and limitations, my students love coming to school and come eager to learn and explore.Have
you ever felt like you had ants in your pants and you needed to groove and move as you were in a meetin
g? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobbl
e chairs are the answer and I love then because they develop their core, which enhances gross motor and
in Turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to sit and do w
orksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our suc
cess. The number toss and color and shape mats can make that happen. My students will forget they are do
ing work and just have the fun a 6 year old deserves.
==================================================

The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teac
her inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-Americ
an, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2% Af
rican-American students. Most of the students are on free or reduced lunch. We aren't receiving doctors,
lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring min
ds of young children and we focus not only on academics but one smart, effective, efficient, and discipl
ined students with good character.In our classroom we can utilize the Bluetooth for swift transitions du
ring class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the vo
lume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as mea
ningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause and repla
y it at any time.\r\nThe cart will allow me to have more room for storage of things that are needed for
the day and has an extra part to it I can use.  The table top chart has all of the letter, words and pic
tures for students to learn about different letters and it is more accessible.
==================================================
```

In [17]:

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

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

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

In [18]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive del
ays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest
working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I
teach in a Title I school where most of the students receive free or reduced price lunch.  Despite their
disabilities and limitations, my students love coming to school and come eager to learn and explore.Have
you ever felt like you had ants in your pants and you needed to groove and move as you were in a meetin
g? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobbl
e chairs are the answer and I love then because they develop their core, which enhances gross motor and
in Turn fine motor skills. \r\nThey also want to learn through games, my kids do not want to sit and do
worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our su
ccess. The number toss and color and shape mats can make that happen. My students will forget they are d
oing work and just have the fun a 6 year old deserves.
==================================================
```

In [19]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive del
ays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest 
working past their limitations.     The materials we have are the ones I seek out for my students. I tea
ch in a Title I school where most of the students receive free or reduced price lunch.  Despite their di
sabilities and limitations, my students love coming to school and come eager to learn and explore.Have y
ou ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting?
This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble c
hairs are the answer and I love then because they develop their core, which enhances gross motor and in 
Turn fine motor skills.   They also want to learn through games, my kids do not want to sit and do works
heets. They want to learn to count by jumping and playing. Physical engagement is the key to our succes
s. The number toss and color and shape mats can make that happen. My students will forget they are doing
work and just have the fun a 6 year old deserves.
```

In [20]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitive dela
ys gross fine motor delays to autism They are eager beavers and always strive to work their hardest work
ing past their limitations The materials we have are the ones I seek out for my students I teach in a Ti
tle I school where most of the students receive free or reduced price lunch Despite their disabilities a
nd limitations my students love coming to school and come eager to learn and explore Have you ever felt 
like you had ants in your pants and you needed to groove and move as you were in a meeting This is how m
y kids feel all the time The want to be able to move as they learn or so they say Wobble chairs are the 
answer and I love then because they develop their core which enhances gross motor and in Turn fine motor
skills They also want to learn through games my kids do not want to sit and do worksheets They want to l
earn to count by jumping and playing Physical engagement is the key to our success The number toss and c
olor and shape mats can make that happen My students will forget they are doing work and just have the f
un a 6 year old deserves
```

In [21]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'the
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', '
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while',
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'befor
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'aga
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each',
            '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',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", '
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'w
            'won', "won't", 'wouldn', "wouldn't"]
```

In [22]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████| 109248/109248 [01:06<00:0
0, 1646.38it/s]
```

In [23]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[23]:
```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gross fine
motor delays autism they eager beavers always strive work hardest working past limitations the materials
ones i seek students i teach title i school students receive free reduced price lunch despite disabiliti
es limitations students love coming school come eager learn explore have ever felt like ants pants neede
d groove move meeting this kids feel time the want able move learn say wobble chairs answer i love devel
op core enhances gross motor turn fine motor skills they also want learn games kids not want sit workshe
ets they want learn count jumping playing physical engagement key success the number toss color shape ma
ts make happen my students forget work fun 6 year old deserves'
```

1.4 Preprocessing of `project_title`

In [24]:
```
# similarly you can preprocess the titles also
project_data.head(2)
```

Out[24]:

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

In [25]:
```
# printing some random project titles.
print(project_data['project_title'].values[54])
print("="*50)
print(project_data['project_title'].values[89])
print("="*50)
print(project_data['project_title'].values[999])
print("="*50)
print(project_data['project_title'].values[11156])
print("="*50)
print(project_data['project_title'].values[89436])
print("="*50)
```

```
Swim For Life At YMCA!
==================================================
Education Through Technology
==================================================
Focus Pocus
==================================================
Making Math Interactive!
==================================================
Classroom Supplies: Help a New Teacher Organize the Classroom!
==================================================
```

In [26]:
```
#Removing phrases from the title features
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"Gotta", "Got to", 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 [27]:
```
#Checkingt titles after removing phrases
sent = decontracted(project_data['project_title'].values[89436])
print(sent)
print("="*50)
```

```
Classroom Supplies: Help a New Teacher Organize the Classroom!
==================================================
```

In [28]:
```
# Remove \\r \\n \\t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
Classroom Supplies: Help a New Teacher Organize the Classroom!
```

In [29]:
```
#Removing numbers & symbols form the titles
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
Classroom Supplies Help a New Teacher Organize the Classroom
```

In [30]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself',
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'the
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', '
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do'
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while'
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'befor
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'aga
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each',
            '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'
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", '
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'w
            'won', "won't", 'wouldn', "wouldn't"]
```

In [31]:

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

```
100%|████████████████████████████████████████████████████████████████████████| 109248/109248 [00:03<00:0
0, 31378.69it/s]
```

In [32]:

```
#checking cleaned text after preprocesing
print(preprocessed_titles[54])
print("="*50)
print(preprocessed_titles[89])
print("="*50)
print(preprocessed_titles[999])
print("="*50)
print(preprocessed_titles[11156])
print("="*50)
print(preprocessed_titles[89436])
```

```
swim for life at ymca
==================================================
education through technology
==================================================
focus pocus
==================================================
making math interactive
==================================================
classroom supplies help new teacher organize classroom
```

1.5 Preparing data for models

In [ ]:

```
project_data.columns
```

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
```

1.5.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/ (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/)

In [ ]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

In [ ]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

In [33]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
#Converting states text into smaller case
project_data['school_state'] = project_data['school_state'].str.lower()
project_data['school_state'].value_counts()
```

Out[33]:

```
ca    15388
tx     7396
ny     7318
fl     6185
nc     5091
il     4350
ga     3963
sc     3936
mi     3161
pa     3109
in     2620
mo     2576
oh     2467
la     2394
ma     2389
wa     2334
ok     2276
nj     2237
az     2147
va     2045
wi     1827
al     1762
ut     1731
tn     1688
ct     1663
md     1514
nv     1367
ms     1323
ky     1304
or     1242
mn     1208
co     1111
ar     1049
id      693
ia      666
ks      634
nm      557
dc      516
hi      507
me      505
wv      503
nh      348
ak      345
de      343
ne      309
sd      300
ri      285
mt      245
nd      143
wy       98
vt       80
Name: school_state, dtype: int64
```

In [ ]:

```
# Applying count vectorizer on school state feature & one hot encoding School_state feature
vectorizer = CountVectorizer(binary=True)
school_state_count = vectorizer.fit_transform(project_data['school_state'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",school_state_count.shape)
```

In [34]:

```
#Replacing spaces & hyphens in the text of project grade category with underscore
#converting Capital letters in the string to smaller letters
#Performing avalue count of project grade category
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-onp
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

Out[34]:

```
grades_prek_2    44225
grades_3_5       37137
grades_6_8       16923
grades_9_12      10963
Name: project_grade_category, dtype: int64
```

In [ ]:

```
#One hot encoding project grade category feature
vectorizer = CountVectorizer(binary=True)
project_grade_one = vectorizer.fit_transform(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encoding ",project_grade_one.shape)
```

In [35]:

```
# check if we have any nan values are there in the column
print(project_data['teacher_prefix'].isnull().values.any())
print("number of nan values",project_data['teacher_prefix'].isnull().values.sum())
```

```
True
number of nan values 3
```

In [36]:

```
#Replacing the Nan values with most frequent value in the column
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
```

In [37]:

```
# check if we have any nan values are there in the column
print(project_data['teacher_prefix'].isnull().values.any())
print("number of nan values",project_data['teacher_prefix'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [38]:

```
#Converting teacher prefix text into smaller case
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.lower()
project_data['teacher_prefix'].value_counts()
```

Out[38]:

```
mrs.       57272
ms.        38955
mr.        10648
teacher     2360
dr.           13
Name: teacher_prefix, dtype: int64
```

In [39]:

```
project_data.isnull().any(axis=0)
```

Out[39]:

```
Unnamed: 0                                      False
id                                              False
teacher_id                                      False
teacher_prefix                                  False
school_state                                    False
project_submitted_datetime                      False
project_grade_category                          False
project_title                                   False
project_essay_1                                 False
project_essay_2                                 False
project_essay_3                                 False
project_essay_4                                 False
project_resource_summary                        False
teacher_number_of_previously_posted_projects    False
project_is_approved                             False
clean_categories                                False
clean_subcategories                             False
essay                                           False
dtype: bool
```

In [ ]:

```
#One hot encoding the teacher prefix column
vectorizer = CountVectorizer(binary=True)
teacher_prefix_one = vectorizer.fit_transform(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",teacher_prefix_one.shape)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [ ]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

In [ ]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",title_bow.shape)
```

1.5.2.2 TFIDF vectorizer

In [ ]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

In [ ]:

```
# you can vectorize the title also
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
title_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",title_tfidf.shape)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [ ]:

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

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

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

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

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

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

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

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


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

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


'''
```

In [40]:

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

In [ ]:

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

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [ ]:

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

In [ ]:

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

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

In [ ]:

```
# Similarly you can vectorize for title also
# Similarly you can vectorize for title also
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_titles = set(tfidf_model.get_feature_names())
```

In [ ]:

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

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

1.5.3 Vectorizing Numerical features

In [41]:

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

In [42]:

```
project_data.columns
```

Out[42]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price',
       'quantity'],
      dtype='object')
```

In [43]:

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

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
```

In [44]:

```
price_standardized
```

Out[44]:

```
array([[-0.3905327 ],
       [ 0.00239637],
       [ 0.59519138],
       ...,
       [-0.15825829],
       [-0.61243967],
       [-0.51216657]])
```

1.5.4 Merging all the above features

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

In [ ]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

In [ ]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

In [45]:

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

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     C:\Users\hims1\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

Out[45]:

```
True
```

__ Computing Sentiment Scores__

In [46]:

```
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
for learning my students learn in many different ways using all of our senses and multiple intelligences i
of techniques to help all my students succeed students in my class come from a variety of different backgr
for wonderful sharing of experiences and cultures including native americans our school is a caring commun
learners which can be seen through collaborative student project based learning in and out of the classroo
in my class love to work with hands on materials and have many different opportunities to practice a skill
mastered having the social skills to work cooperatively with friends is a crucial aspect of the kindergart
montana is the perfect place to learn about agriculture and nutrition my students love to role play in our
in the early childhood classroom i have had several kids ask me can we try cooking with real food i will t
and create common core cooking lessons where we learn important math and writing concepts while cooking de
food for snack time my students will have a grounded appreciation for the work that went into making the f
of where the ingredients came from as well as how it is healthy for their bodies this project would expand
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce ma
and mix up healthy plants from our classroom garden in the spring we will also create our own cookbooks to
shared with families students will gain math and literature skills as well as a life long enjoyment for he
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,
```

Assignment 5: Logistic Regression

1. **[Task-1] Logistic Regression(either SGDClassifier with log loss, or LogisticRegression) on these feature sets**

    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (`BOW with bi-grams` with `min_df=10` and `max_features=5000`)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (`TFIDF with bi-grams` with `min_df=10` and `max_features=5000`)
    - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)

2. **Hyper paramter tuning (find best hyper parameters corresponding the algorithm that you choose)**

- Find the best hyper parameter which will give the maximum [AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
- Find the best hyper paramter using k-fold cross validation or simple cross validation data
- Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Representation of results**

   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure.
   - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
   - Along with plotting ROC curve, you need to print the [confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using [seaborn heatmaps. (https://seaborn.pydata.org/generated/seaborn.heatmap.html)](https://seaborn.pydata.org/generated/seaborn.heatmap.html)

4. **[Task-2] Apply Logistic Regression on the below feature set Set 5 by finding the best hyper parameter as suggested in step 2 and step 3.**
5. Consider these set of features Set 5 :

   - **school_state** : categorical data
   - **clean_categories** : categorical data
   - **clean_subcategories** : categorical data
   - **project_grade_category** :categorical data
   - **teacher_prefix** : categorical data
   - **quantity** : numerical data
   - **teacher_number_of_previously_posted_projects** : numerical data
   - **price** : numerical data
   - **sentiment score's of each of the essay** : numerical data
   - **number of words in the title** : numerical data
   - **number of words in the combine essays** : numerical data

   And apply the Logistic regression on these features by finding the best hyper paramter as suggested in step 2 and step 3

6. **Conclusion**

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

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakage, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this [link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)](https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2. Logistic Regression

2.1 Splitting data into Train and cross validation(or test): Stratified Sampling

In [47]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
data = project_data
data.head(5)
```

Out[47]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs.	in	2016-12-05 13:43:57	grade
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr.	fl	2016-10-25 09:22:10	gr
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	ms.	az	2016-08-31 12:03:56	gr
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	mrs.	ky	2016-10-06 21:16:17	grade
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	mrs.	tx	2016-07-11 01:10:09	grade

In [48]:

```
data.shape
```

Out[48]:

```
(109248, 20)
```

In [49]:

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

Out[49]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_c
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs.	in	2016-12-05 13:43:57	grades

In [50]:

```
X = data
```

In [51]:

```
# check if we have any nan values are there in the column
print(X['teacher_prefix'].isnull().values.any())
print("number of nan values",X['teacher_prefix'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [52]:

```
#Replacing the Nan values with most frequent value in the column
X['teacher_prefix']=X['teacher_prefix'].fillna('Mrs.')
```

In [53]:

```
#Converting teacher prefix text into smaller case
X['teacher_prefix'] = X['teacher_prefix'].str.lower()
X['teacher_prefix'].value_counts()
```

Out[53]:

```
mrs.       57272
ms.        38955
mr.        10648
teacher     2360
dr.           13
Name: teacher_prefix, dtype: int64
```

Splitting data into Train and cross validation(or test): Stratified Sampling

In [54]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
#Splitting data into test & train set
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.33,stratify=y)
```

In [55]:

```
#Splitting training data into training & cross validation sets
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train,
                                                stratify= y_train,
                                                test_size = 0.33)
```

1.3 Text preprocessing

In [56]:
```
# printing some random reviews
print(X_train['essay'].values[0])
print("="*50)
print(X_train['essay'].values[100])
print("="*50)
print(X_train['essay'].values[300])
print("="*50)
print(X_train['essay'].values[5000])
print("="*50)
print(X_train['essay'].values[20000])
print("="*50)
```

```
Do you remember the first time that you had a teacher who made learning fun? This is what I strive to
do every day in my classroom. By incorporating STEM into my students' everyday learning, my students a
re able to learn while also exploring their creative abilities.\r\n\r\nI teach an amazing group of fir
st graders who love to learn.  They are self-motivated and always up for any academic challenge, espec
ially when it involves hands-on opportunities. I want to support and foster their love of learning by
incorporating  engaging, creative activities that support our curriculum.  My students need STEM mater
ials to help them learn using hands-on manipulatives and building materials. \r\n\r\n\r\nOur project i
s aimed at providing students with Science, Technology, Engineering, and Math materials.\r\n\r\nIt is
never too early to engage students in STEM.   My first graders are eager to learn and explore these su
bject areas.\r\nThe requested materials will help my students develop a deeper understanding of math a
nd science tasks by providing them with manipulatives to solve problems. These supplies will spark ima
gination and creativity through the use of building, problem solving, and exploring with hands-on lear
ning.\r\n\r\nThe goal of this project is to engage my students and make learning FUN! These STEM and E
ngineering kits will help to keep my young students motivated and eager to learn!
==================================================
My students are all English Language Learners that come from low income homes.  ALL of our students qu
alify for the free meal program. We are a Title I school as well as a Performance Improvement school.
Many of my students are struggling to learn, understand, and connect concepts. \r\nTo ease the struggl
e, your ink donations will provide colors to their creative work and enhance their social and academic
```

In [57]:
```
# 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 [58]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive del
ays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest
working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I
teach in a Title I school where most of the students receive free or reduced price lunch.  Despite their
disabilities and limitations, my students love coming to school and come eager to learn and explore.Have
you ever felt like you had ants in your pants and you needed to groove and move as you were in a meetin
g? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobbl
e chairs are the answer and I love then because they develop their core, which enhances gross motor and
in Turn fine motor skills. \r\nThey also want to learn through games, my kids do not want to sit and do
worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our su
ccess. The number toss and color and shape mats can make that happen. My students will forget they are d
oing work and just have the fun a 6 year old deserves.
==================================================
```

In [59]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive del
ays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest
working past their limitations.     The materials we have are the ones I seek out for my students. I tea
ch in a Title I school where most of the students receive free or reduced price lunch.  Despite their di
sabilities and limitations, my students love coming to school and come eager to learn and explore.Have y
ou ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting?
This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble c
hairs are the answer and I love then because they develop their core, which enhances gross motor and in
Turn fine motor skills.   They also want to learn through games, my kids do not want to sit and do works
heets. They want to learn to count by jumping and playing. Physical engagement is the key to our succes
s. The number toss and color and shape mats can make that happen. My students will forget they are doing
work and just have the fun a 6 year old deserves.
```

In [60]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitive dela
ys gross fine motor delays to autism They are eager beavers and always strive to work their hardest work
ing past their limitations The materials we have are the ones I seek out for my students I teach in a Ti
tle I school where most of the students receive free or reduced price lunch Despite their disabilities a
nd limitations my students love coming to school and come eager to learn and explore Have you ever felt
like you had ants in your pants and you needed to groove and move as you were in a meeting This is how m
y kids feel all the time The want to be able to move as they learn or so they say Wobble chairs are the
answer and I love then because they develop their core which enhances gross motor and in Turn fine motor
skills They also want to learn through games my kids do not want to sit and do worksheets They want to l
earn to count by jumping and playing Physical engagement is the key to our success The number toss and c
olor and shape mats can make that happen My students will forget they are doing work and just have the f
un a 6 year old deserves
```

In [61]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'the
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', '
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do'
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while'
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'befor
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'aga
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each',
            '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'
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", '
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'we
            'won', "won't", 'wouldn', "wouldn't"]
```

Preprocessing for Train Data

```
In [62]: # Combining all the above stundents
         from tqdm import tqdm
         preprocessed_essays_xtr = []
         # 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_xtr.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 49041/49041 [00:29<00:0
0, 1660.37it/s]
```

```
In [63]: # after preprocesing
         preprocessed_essays_xtr[300]
```

```
Out[63]: 'school students come variety backgrounds cultures school serves 1 024 students one seven high schools c
ounty school district lies edge greater washington dc area rural parts southern maryland although studen
ts come diverse backgrounds include best brightest successful students area school motto pride excellenc
e education preparation respect integrity determination excellence students demonstrate qualities every
day however students difficulty applying classroom instruction real world scenarios hopes bring real wor
ld connections school students modern technology virtual reality glasses give students opportunity apply
geometric concepts real world historical locations virtual reality glasses students break free classroom
walls visit historical locations use measuring tool clinometer calculate angles locations apply indirect
measurement methods calculate heights distances historical sites without ever leaving classroom although
plan use virtual reality glasses students also made available teachers school media center teachers able
check virtual reality glasses use within content area virtual reality glasses would allow students oppor
tunities go virtual field trips experiment 360 videos pictures learn course content new exciting ways al
so plan offer assistance professional development opportunity fellow teachers use virtual reality glasse
s content areas'
```

```
In [64]: # Combining all the above stundents
         from tqdm import tqdm
         preprocessed_essays_xcv = []
         # 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_xcv.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 24155/24155 [00:14<00:0
0, 1700.77it/s]
```

```
In [65]: # after preprocesing
         preprocessed_essays_xcv[300]
```

```
Out[65]: 'small rural school district strives make learning enjoyable meaningful experience students mission ensu
re child reaches full potential incorporating wealth resources learning opportunities currently transfor
ming kindergarten first second grade classrooms 21st century learning environments one obstacle met enou
gh flexible seating options meet individual needs students comfortable sitting behind desk no well stude
nts not either chances not comfortable unlikely fully engaged 21st century students learn differently tr
ansforming kindergarten first second grade classrooms 21st century learning environments students flexib
ility choose seat comfortable inviting flexible seating allows students discover learn best helps focus
stay engaged optimal learning growth big comfy pillows add flexible seating options allow students get c
omfortable work'
```

In [66]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_xte = []
# 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_xte.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 36052/36052 [00:21<00:0
0, 1679.33it/s]
```

In [67]:

```
# after preprocesing
preprocessed_essays_xte[300]
```

Out[67]:

```
'students middle school students age eleven fourteen dedicate school day school day even weekends school
music program conduct five different ensembles school including concert band jazz band marching band par
ticipate two annual music competitions one statewide one northeast regional also perform local elementar
y schools spring musical addition performing classes students core music program well students learn ess
entials foundations music reading notation counting rhythms playing keyboard even performing bucket drum
s teach students composers time periods music including baroque classical romantic contemporary jazz age
important learning men women understand much possible includes visual image learning best print photos p
rinter black white therefore students partial representation individuals color laser printer able print
quality images students accurate representation musical masters'
```

1.4 Preprocessing of `project_title`

In [68]:

```
# similarly you can preprocess the titles also
#printing random titles
print(data['project_title'].values[49])
print("="*50)
print(data['project_title'].values[89])
print("="*50)
print(data['project_title'].values[999])
print("="*50)
print(data['project_title'].values[11156])
print("="*50)
print(data['project_title'].values[20000])
print("="*50)
```

```
Rainy Day Run Around!
==================================================
Education Through Technology
==================================================
Focus Pocus
==================================================
Making Math Interactive!
==================================================
We Need To Move It While We Input It!
==================================================
In [69]: #Removing phrases from the title features
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)
    phrase = re.sub(r"Gotta",  "Got to",  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 [70]: #Checkingt titles after removing phrases
         sent = decontracted(project_data['project_title'].values[89436])
         print(sent)
         print("="*50)
```

```
Classroom Supplies: Help a New Teacher Organize the Classroom!
==================================================
```

```
In [71]: # Remove \\r \\n \\t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
         sent = sent.replace('\\r', ' ')
         sent = sent.replace('\\"', ' ')
         sent = sent.replace('\\n', ' ')
         print(sent)
```

```
Classroom Supplies: Help a New Teacher Organize the Classroom!
```

```
In [72]: #Removing stop words from the preprocessed titles
         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', 'the
                     'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', '
                     'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do'
                     'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while'
                     'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'befor
                     'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'aga
                     'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each',
                     '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'
                     've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't
                     "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", '
                     "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'we
                     'won', "won't", 'wouldn', "wouldn't"]
```

```
In [73]: preprocessed_titles_xtr = []
         # tqdm is for printing the status bar
         for sentance in tqdm(X_train['project_title'].values):
             sent = decontracted(sentance)
             sent = sent.replace('\\r', ' ')
             sent = sent.replace('\\"', ' ')
             sent = sent.replace('\\n', ' ')
             sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
             # https://gist.github.com/sebleier/554280
             sent = ' '.join(e for e in sent.split() if e not in stopwords)
             preprocessed_titles_xtr.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 49041/49041 [00:01<00:0
0, 31060.40it/s]
```

In [74]:

```
#checking cleaned text after preprocesing
print(preprocessed_titles_xtr[89])
print("="*50)
```

```
teaching healthy eating through cooking club
==================================================
```

In [75]:

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

```
100%|████████████████████████████████████████████████████████████████████████| 24155/24155 [00:00<00:0
0, 26985.31it/s]
```

In [76]:

```
print(preprocessed_titles_xcv[89])
print("="*50)
```

```
code learning
==================================================
```

In [77]:

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

```
100%|████████████████████████████████████████████████████████████████████████| 36052/36052 [00:01<00:0
0, 28605.88it/s]
```

In [78]:

```
print(preprocessed_titles_xte[89])
print("="*50)
```

```
make every moment count
==================================================
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [79]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
#We use fit only for train data
vectorizer_state = CountVectorizer(binary=True)
vectorizer_state.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer_state.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer_state.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer_state.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer_state.get_feature_names())
print("="*75)
```

```
After vectorizations
(49041, 51) (49041,)
(24155, 51) (24155,)
(36052, 51) (36052,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks',
'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'n
y', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
===========================================================================
```

2.2.2 One hot encoding the categorical features : teacher_prefix

In [80]:

```
# check if we have any nan values are there in the column
print(X_train['teacher_prefix'].isnull().values.any())
print("number of nan values",X_train['teacher_prefix'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [81]:

```
# check if we have any nan values are there in the column
print(X_cv['teacher_prefix'].isnull().values.any())
print("number of nan values",X_cv['teacher_prefix'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [82]:

```
# check if we have any nan values are there in the column
print(X_test['teacher_prefix'].isnull().values.any())
print("number of nan values",X_test['teacher_prefix'].isnull().values.sum())
```

```
False
number of nan values 0
```

In [83]:

```
# we use count vectorizer to convert the values into one
#We use fit only for train data
vectorizer_tp = CountVectorizer(binary=True)
vectorizer_tp.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer_tp.transform(X_train['teacher_prefix'].values)
X_cv_teacher_ohe = vectorizer_tp.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer_tp.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer_tp.get_feature_names())
print("="*50)
```

```
After vectorizations
(49041, 5) (49041,)
(24155, 5) (24155,)
(36052, 5) (36052,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
==================================================
```

2.2.3 One hot encoding the categorical features : grades

In [84]:

```
#Replacing spaces & hyphens in the text of project grade category with underscore
#converting Capital letters in the string to smaller letters
#Performing avalue count of project grade category
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-on-
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

Out[84]:

```
grades_prek_2    44225
grades_3_5       37137
grades_6_8       16923
grades_9_12      10963
Name: project_grade_category, dtype: int64
```

In [85]:

```
#We use fit only for train data
vectorizer_grade = CountVectorizer()
vectorizer_grade.fit(X_train['project_grade_category'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer_grade.transform(X_train['project_grade_category'].values)
X_cv_grade_ohe = vectorizer_grade.transform(X_cv['project_grade_category'].values)
X_test_grade_ohe = vectorizer_grade.transform(X_test['project_grade_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer_grade.get_feature_names())
print("="*70)
```

```
After vectorizations
(49041, 4) (49041,)
(24155, 4) (24155,)
(36052, 4) (36052,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
======================================================================
```

2.2.4 One hot encoding the categorical features : project subject category

In [86]:
```
#We use fit only for train data
vectorizer_category = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), binary=True)
vectorizer_category.fit(X_train['clean_categories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_cat_ohe = vectorizer_category.transform(X_train['clean_categories'].values)
X_cv_cat_ohe = vectorizer_category.transform(X_cv['clean_categories'].values)
X_test_cat_ohe = vectorizer_category.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_cat_ohe.shape, y_train.shape)
print(X_cv_cat_ohe.shape, y_cv.shape)
print(X_test_cat_ohe.shape, y_test.shape)
print(vectorizer_category.get_feature_names())
print("="*70)
```

```
After vectorizations
(49041, 9) (49041,)
(24155, 9) (24155,)
(36052, 9) (36052,)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Spo
rts', 'Math_Science', 'Literacy_Language']
======================================================================
```

In [87]:
```
#We use fit only for train data
vectorizer_subcat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), binary=True)
vectorizer_subcat.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_subcat_ohe = vectorizer_subcat.transform(X_train['clean_subcategories'].values)
X_cv_subcat_ohe = vectorizer_subcat.transform(X_cv['clean_subcategories'].values)
X_test_subcat_ohe = vectorizer_subcat.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_subcat_ohe.shape, y_train.shape)
print(X_cv_subcat_ohe.shape, y_cv.shape)
print(X_test_subcat_ohe.shape, y_test.shape)
print(vectorizer_subcat.get_feature_names())
print("="*70)
```

```
After vectorizations
(49041, 30) (49041,)
(24155, 30) (24155,)
(36052, 30) (36052,)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_G
overnment', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'Perfor
mingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geograph
y', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArt
s', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literac
y']
======================================================================
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [88]:
```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separtly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

i) BoW encoding

1.5.2.1 Bag of words on Essay Feature

In [89]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
#Applying BoW on essays feature
#Considering only the words which appear atleast in 10 documents or reviews
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)



from sklearn.feature_extraction.text import CountVectorizer
vectorizer_essay_bow = CountVectorizer(ngram_range=(1, 2), min_df=10, max_features=5000)
vectorizer_essay_bow.fit(preprocessed_essays_xtr) # fiting only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer_essay_bow.transform(preprocessed_essays_xtr)
X_cv_essay_bow = vectorizer_essay_bow.transform(preprocessed_essays_xcv)
X_test_essay_bow = vectorizer_essay_bow.transform(preprocessed_essays_xte)

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

```
(49041, 19) (49041,)
(24155, 19) (24155,)
(36052, 19) (36052,)
====================================================================================================
After vectorizations
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
====================================================================================================
```

1.5.2.2 Bag of words on Project Title feature

In [90]:

```python
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
#Applying BoW on project titles feature
#Considering only the words which appear atleast in 10 documents or reviews
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)



from sklearn.feature_extraction.text import CountVectorizer
vectorizer_titles_bow = CountVectorizer(ngram_range=(1, 2), min_df=10, max_features=5000)
vectorizer_titles_bow.fit(preprocessed_titles_xtr) # fiting only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_titles_bow = vectorizer_titles_bow.transform(preprocessed_titles_xtr)
X_cv_titles_bow = vectorizer_titles_bow.transform(preprocessed_titles_xcv)
X_test_titles_bow = vectorizer_titles_bow.transform(preprocessed_titles_xte)

print("After vectorizations")
print(X_train_titles_bow.shape, y_train.shape)
print(X_cv_titles_bow.shape, y_cv.shape)
print(X_test_titles_bow.shape, y_test.shape)
print("="*100)
```

```
(49041, 19) (49041,)
(24155, 19) (24155,)
(36052, 19) (36052,)
====================================================================================================
After vectorizations
(49041, 3772) (49041,)
(24155, 3772) (24155,)
(36052, 3772) (36052,)
====================================================================================================
```

ii) TFIDF Vectorization

TFIDF vectorizer on essay feature

In [91]:

```
#Applying TF-IDF on essays feature
#Considering only the words which appear atleast in 10 documents or reviews
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)



from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_essay_tfidf = TfidfVectorizer(ngram_range=(1, 2), min_df=10, max_features=5000)
vectorizer_essay_tfidf.fit(preprocessed_essays_xtr) # fiting only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf = vectorizer_essay_tfidf.transform(preprocessed_essays_xtr)
X_cv_essay_tfidf = vectorizer_essay_tfidf.transform(preprocessed_essays_xcv)
X_test_essay_tfidf = vectorizer_essay_tfidf.transform(preprocessed_essays_xte)

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

```
(49041, 19) (49041,)
(24155, 19) (24155,)
(36052, 19) (36052,)
====================================================================================================
After vectorizations
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
====================================================================================================
```

TFIDF on Project Title feature

In [92]:

```
#Applying Tfidf on project titles feature
#Considering only the words which appear atleast in 10 documents or reviews
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)


from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_title = TfidfVectorizer(ngram_range=(1, 2), min_df=10, max_features=5000)
vectorizer_tfidf_title.fit(preprocessed_titles_xtr) # fiting only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_titles_tfidf = vectorizer_tfidf_title.transform(preprocessed_titles_xtr)
X_cv_titles_tfidf = vectorizer_tfidf_title.transform(preprocessed_titles_xcv)
X_test_titles_tfidf = vectorizer_tfidf_title.transform(preprocessed_titles_xte)

print("After vectorizations")
print(X_train_titles_tfidf.shape, y_train.shape)
print(X_cv_titles_tfidf.shape, y_cv.shape)
print(X_test_titles_tfidf.shape, y_test.shape)
print("="*100)
```

```
(49041, 19) (49041,)
(24155, 19) (24155,)
(36052, 19) (36052,)
====================================================================================================
After vectorizations
(49041, 3772) (49041,)
(24155, 3772) (24155,)
(36052, 3772) (36052,)
====================================================================================================
```

iii) Using Pretrained Models : AvgW2V

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

In [ ]:
```
model = loadGloveModel('glove.42B.300d.txt')
```

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

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

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

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

In [ ]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-l

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

In [93]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-l

with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words = set(model.keys())
```

Applying to Train set for Essay feature

In [94]:
```
preprocessed_essays_xtr[0]
```

Out[94]:
```
'remember first time teacher made learning fun strive every day classroom incorporating stem students ev
eryday learning students able learn also exploring creative abilities teach amazing group first graders
love learn self motivated always academic challenge especially involves hands opportunities want support
foster love learning incorporating engaging creative activities support curriculum students need stem ma
terials help learn using hands manipulatives building materials project aimed providing students science
technology engineering math materials never early engage students stem first graders eager learn explore
subject areas requested materials help students develop deeper understanding math science tasks providin
g manipulatives solve problems supplies spark imagination creativity use building problem solving explor
ing hands learning goal project engage students make learning fun stem engineering kits help keep young
students motivated eager learn'
```

In [95]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_extr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_xtr): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_extr.append(vector)

print(len(avg_w2v_vectors_extr))
print(len(avg_w2v_vectors_extr[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 49041/49041 [00:20<00:0
0, 2446.07it/s]

49041
300
```

Applying to Cross validation set for Essay feature

In [96]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_excv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_xcv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_excv.append(vector)

print(len(avg_w2v_vectors_excv))
print(len(avg_w2v_vectors_excv[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 24155/24155 [00:09<00:0
0, 2448.44it/s]

24155
300
```

Applying to test set for Essay feature

In [97]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_exte = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_xte): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_exte.append(vector)

print(len(avg_w2v_vectors_exte))
print(len(avg_w2v_vectors_exte[0]))
```

```
100%|█████████████████████████████████████████████████████████████████████████| 36052/36052 [00:14<00:0
0, 2463.90it/s]

36052
300
```

Applying to Train set for Project title feature

In [98]:

```
# Vectorizing project_title using avgw2v method
avg_w2v_vectors_txtr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_xtr): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_txtr.append(vector)

print(len(avg_w2v_vectors_txtr))
print(len(avg_w2v_vectors_txtr[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████| 49041/49041 [00:01<00:0
0, 45712.96it/s]

49041
300
```

Applying to Cross validation set for Project title feature

In [99]:

```
# Vectorizing project_title using avgw2v method
avg_w2v_vectors_txcv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_xcv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_txcv.append(vector)

print(len(avg_w2v_vectors_txcv))
print(len(avg_w2v_vectors_txcv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████| 24155/24155 [00:00<00:0
0, 41805.02it/s]

24155
300
```

Applying to Test set for Project title feature

In [100]:

```
# Vectorizing project_title using avgw2v method
avg_w2v_vectors_txte = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_xte): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_txte.append(vector)

print(len(avg_w2v_vectors_txte))
print(len(avg_w2v_vectors_txte[0]))
```

```
100%|█████████████████████████████████████████████████████████████████████████| 36052/36052 [00:00<00:0
0, 44303.58it/s]

36052
300
```

iv) Using Pretrained Models: TFIDF weighted W2V

Applying on Training set of essays feature

In [101]:

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

In [102]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_extr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_xtr): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf val
            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_extr.append(vector)

print(len(tfidf_w2v_vectors_extr))
print(len(tfidf_w2v_vectors_extr[0]))
```

```
100%|███████████████████████████████████████████████████████████████████████████| 49041/49041 [02:04<00:
00, 393.79it/s]

49041
300
```

Applying on Cross validation set of essays feature

In [103]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_excv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_xcv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf va
            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_excv.append(vector)

print(len(tfidf_w2v_vectors_excv))
print(len(tfidf_w2v_vectors_excv[0]))
```

```
100%|███████████████████████████████████████████████████████████████████████████| 24155/24155 [01:01<00:
00, 393.05it/s]

24155
300
```

Applying on test set of essays feature

In [104]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_exte = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays_xte): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf va
            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_exte.append(vector)

print(len(tfidf_w2v_vectors_exte))
print(len(tfidf_w2v_vectors_exte[0]))
```

```
100%|███████████████████████████████████████████████████████████████████████████| 36052/36052 [01:32<00:
00, 389.44it/s]

36052
300
```

Applying on Training set of project title feature

In [105]:
```
# Similarly you can vectorize for title also
# vectorizing project_title using TFIDF weighted W2V pretrained model
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_titles_xtr)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [106]:
```
# #compute tfidf w2v for project titles in train set

# tfidf_w2v_vectors_txtr = []; # the avg-w2v for each sentence/review is stored in this list

# for sentence in tqdm(preprocessed_titles_xtr): # for each review/sentence in Xtrain
#     vector = np.zeros(300) # as word vectors are of zero length
#     tf_idf_weight =0; # num of words with a valid vector in the sentence/review
#     for word in sentence.split(): # for each word in a review/sentence
#         if word in glove_words and (word in tfidf_words):
#             vector = model[word]

#             tf_idf = dictionary[word]*(sentance.count(word)/len(sentance.split()))
#             vector += (vector * tf_idf)
#             tf_idf_weight += tf_idf
#     if tf_idf_weight != 0:
#         vector /= tf_idf_weight
#     tfidf_w2v_vectors_txtr.append(vector)

# print(len(tfidf_w2v_vectors_txtr))
# print(len(tfidf_w2v_vectors_txtr[0]))
```

In [107]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_txtr = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_xtr): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf va
            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_txtr.append(vector)

print(len(tfidf_w2v_vectors_txtr))
print(len(tfidf_w2v_vectors_txtr[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 49041/49041 [00:02<00:0
0, 21667.23it/s]

49041
300
```

Applying on Cross validation set of project title feature

In [108]:
```
# #compute tfidf w2v for project titles in cv set

# tfidf_w2v_vectors_txcv = []; # the avg-w2v for each sentence/review is stored in this list

# for sentence in tqdm(preprocessed_titles_xcv): # for each review/sentence in Xtrain
#     vector = np.zeros(300) # as word vectors are of zero length
#     tf_idf_weight =0; # num of words with a valid vector in the sentence/review
#     for word in sentence.split(): # for each word in a review/sentence
#         if word in glove_words and (word in tfidf_words):
#             vector = model[word]

#             tf_idf = dictionary[word]*(sentance.count(word)/len(sentance.split()))
#             vector += (vector * tf_idf)
#             tf_idf_weight += tf_idf
#     if tf_idf_weight != 0:
#         vector /= tf_idf_weight
#     tfidf_w2v_vectors_txcv.append(vector)

# print(len(tfidf_w2v_vectors_txcv))
# print(len(tfidf_w2v_vectors_txcv[0]))
```

In [109]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_txcv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_xcv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf val
            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_txcv.append(vector)

print(len(tfidf_w2v_vectors_txcv))
print(len(tfidf_w2v_vectors_txcv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████| 24155/24155 [00:01<00:0
0, 21543.07it/s]

24155
300
```

Applying on test set of project title feature

In [110]:
```
# #compute tfidf w2v for project titles in test set

# tfidf_w2v_vectors_txte = []; # the avg-w2v for each sentence/review is stored in this list

# for sentence in tqdm(preprocessed_titles_xte): # for each review/sentence in Xtrain
#     vector = np.zeros(300) # as word vectors are of zero length
#     tf_idf_weight =0; # num of words with a valid vector in the sentence/review
#     for word in sentence.split(): # for each word in a review/sentence
#         if word in glove_words and (word in tfidf_words):
#             vector = model[word]

#             tf_idf = dictionary[word]*(sentance.count(word)/len(sentance.split()))
#             vector += (vector * tf_idf)
#             tf_idf_weight += tf_idf
#     if tf_idf_weight != 0:
#         vector /= tf_idf_weight
#     tfidf_w2v_vectors_txte.append(vector)

# print(len(tfidf_w2v_vectors_txte))
# print(len(tfidf_w2v_vectors_txte[0]))
```

In [111]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_txte = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_titles_xte): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/l
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf va
            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_txte.append(vector)

print(len(tfidf_w2v_vectors_txte))
print(len(tfidf_w2v_vectors_txte[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 36052/36052 [00:01<00:0
0, 21723.00it/s]

36052
300
```

1.5.3 Vectorizing Numerical features

For Price feature

In [112]:

```
X_train['price'].shape
```

Out[112]:

```
(49041,)
```

In [113]:

```
from sklearn.preprocessing import Normalizer
price_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.
price_normalizer.fit(X_train['price'].values.reshape(1,-1))

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

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

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

In [114]:

```
X_train_price_norm = X_train_price_norm.T
X_cv_price_norm = X_cv_price_norm.T
X_test_price_norm = X_test_price_norm.T

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

```
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

For Quantity

In [115]:

```
#Normalizing quantity
from sklearn.preprocessing import Normalizer
quan_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.
quan_normalizer.fit(X_train['quantity'].values.reshape(1,-1))

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

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

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

In [116]:

```
X_train_quantity_norm = X_train_quantity_norm.T
X_cv_quantity_norm = X_cv_quantity_norm.T
X_test_quantity_norm = X_test_quantity_norm.T

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

```
Final Matrix
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

For teacher previously posted projects

In [117]:
```
# Normalizing teacher previously posted projects
from sklearn.preprocessing import Normalizer
tpp_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.
tpp_normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

X_train_tpp_norm = tpp_normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values
X_cv_tpp_norm = tpp_normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.resha
X_test_tpp_norm = tpp_normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.r

print("After vectorizations")
print(X_train_tpp_norm.shape, y_train.shape)
print(X_cv_tpp_norm.shape, y_cv.shape)
print(X_test_tpp_norm.shape, y_test.shape)
print("="*100)
```

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

In [118]:
```
X_train_tpp_norm = X_train_tpp_norm.T
X_cv_tpp_norm = X_cv_tpp_norm.T
X_test_tpp_norm = X_test_tpp_norm.T

print(X_train_tpp_norm.shape, y_train.shape)
print(X_cv_tpp_norm.shape, y_cv.shape)
print(X_test_tpp_norm.shape, y_test.shape)
print("="*100)
```

```
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

Merging Numerical & Categorical features

- we need to merge all the numerical vectors & catogorical features

In [119]:
```
merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
rom scipy.sparse import hstack
_tr_numcat = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_price_norm, X_trair
_cv_numcat = hstack((X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_price_norm, X_cv_cat_ohe, X_cv_
_te_numcat = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_price_norm, X_test_cat_

rint("Final Data matrix")
rint(X_tr_numcat.shape, y_train.shape)
rint(X_cv_numcat.shape, y_cv.shape)
rint(X_te_numcat.shape, y_test.shape)
rint("="*100)
```

```
Final Data matrix
(49041, 102) (49041,)
(24155, 102) (24155,)
(36052, 102) (36052,)
====================================================================================================
```

2.4 Appling Logistic Regression on different kind of featurization as mentioned in the instructions

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

In [120]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

2.4.1 Applying Logistic Regression on BOW, SET 1

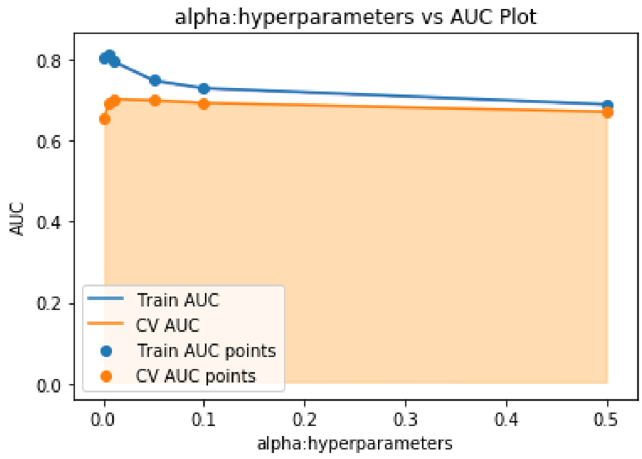
Consider Set 1 :- categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

In [121]:

```
# Please write all the code with proper documentation
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_set1 = hstack((X_train_essay_bow, X_train_titles_bow, X_tr_numcat )).tocsr()
X_cv_set1 = hstack((X_cv_essay_bow, X_cv_titles_bow, X_cv_numcat)).tocsr()
X_te_set1 = hstack((X_test_essay_bow, X_test_titles_bow, X_te_numcat )).tocsr()

print("Final Data matrix")
print(X_tr_set1.shape, y_train.shape)
print(X_cv_set1.shape, y_cv.shape)
print(X_te_set1.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 8874) (49041,)
(24155, 8874) (24155,)
(36052, 8874) (36052,)
====================================================================================================
```

In [122]:
```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Mo
from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression,SGDClassifier
import matplotlib.pyplot as plt

data = data #refer: http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.h

tuned_parameters = {'alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5]}

#Using SGDClassifier
model = GridSearchCV(SGDClassifier(loss='log', class_weight='balanced'), tuned_parameters, cv=5, scoring=
model.fit(X_tr_set1, y_train)

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

plt.figure()
plt.plot(tuned_parameters['alpha'],train_auc,label="Train AUC")
plt.gca().fill_between(tuned_parameters['alpha'],train_auc - train_auc_std,train_auc + train_auc_std,alpha

plt.plot(tuned_parameters['alpha'],cv_auc,label="CV AUC")
plt.gca().fill_between(tuned_parameters['alpha'],cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorang
plt.scatter(tuned_parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(tuned_parameters['alpha'], cv_auc, label='CV AUC points')

plt.legend(loc='best')
plt.xlabel("alpha:hyperparameters")
plt.ylabel("AUC")
plt.title("alpha:hyperparameters vs AUC Plot")
plt.show()
```



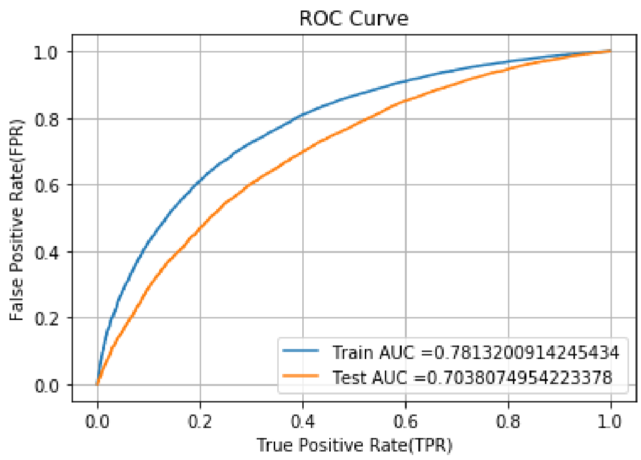
Train the model using best hyper parameter

In [123]:
```
best_a = model.best_params_

best_a = list(best_a.values())[0]
print("Best a :{0}".format(best_a))
print(model.best_score_)
```

```
Best a :0.01
0.7012490698584884
```

In [124]:
```
def batch_predict(clf, data):

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
# consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 49000
# in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
# we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [125]:
```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curv
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression,SGDClassifier

model = SGDClassifier(loss='log', alpha= best_a, class_weight='balanced')

model.fit(X_tr_set1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr_set1)
y_test_pred = batch_predict(model, X_te_set1)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC Curve")
plt.grid(True)
plt.show()
```



In [126]:
```
# we are writing our own function for predict, with defined threshold
# 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.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [127]:

```
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_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_tpr)))
```

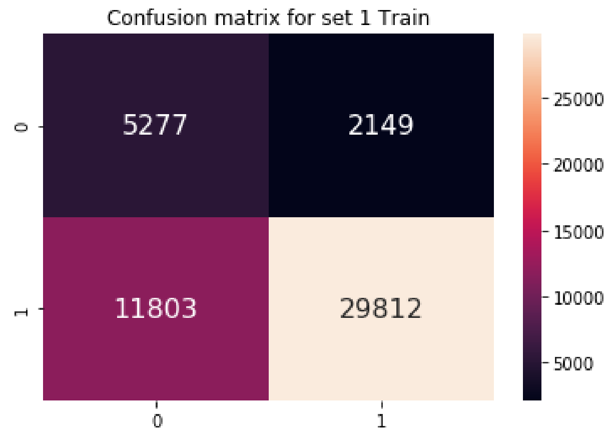
```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5090651454396502 for threshold 0.567
[[ 5277  2149]
 [11803 29812]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.5090651454396502 for threshold 0.567
[[ 3279  2180]
 [ 9279 21314]]
```

In [203]:

```
#How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = [[5277,2149],
        [11803,29812]]
df_cm = pd.DataFrame(array, index = [i for i in "01"],
                  columns = [i for i in "01"])
plt.figure
plt.title('Confusion matrix for set 1 Train')
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

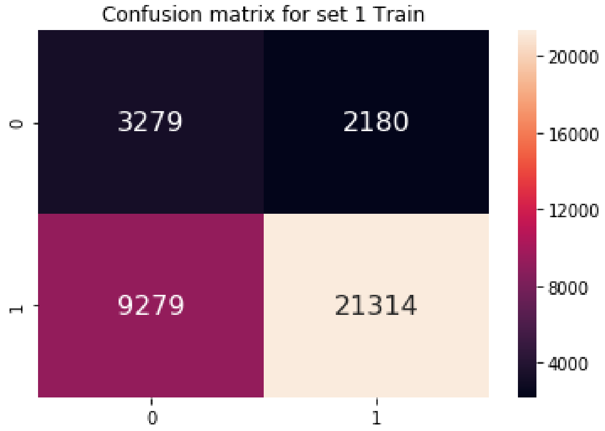
Out[203]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7c6b075c0>
```



In [204]:
```
#How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = [[3279,2180],
        [9279,21314]]
df_cm = pd.DataFrame(array, index = [i for i in "01"],
                  columns = [i for i in "01"])
plt.figure
plt.title('Confusion matrix for set 1 Train')
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

Out[204]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7c1887438>
```



2.4.2 Applying Logistic Regression on TFIDF, SET 2

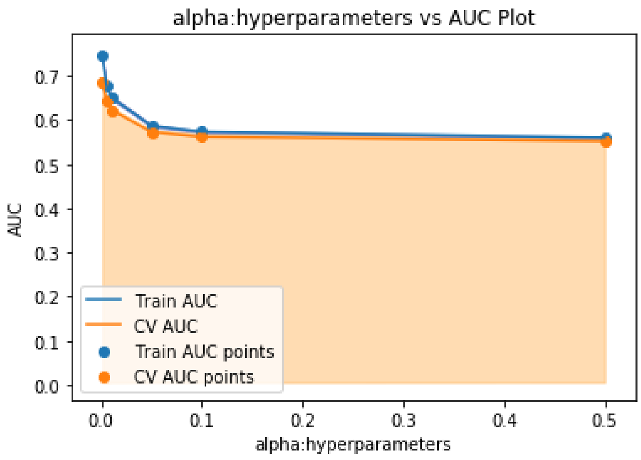
Consider Set 2 :- categorical, numerical features + project_title(TFIDF) + preprocessed_essay (TFIDF)

In [130]:
```
# Please write all the code with proper documentation
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_set2 = hstack((X_train_essay_tfidf, X_train_titles_tfidf, X_tr_numcat )).tocsr()
X_cv_set2 = hstack((X_cv_essay_tfidf, X_cv_titles_tfidf, X_cv_numcat)).tocsr()
X_te_set2 = hstack((X_test_essay_tfidf, X_test_titles_tfidf, X_te_numcat )).tocsr()

print("Final Data matrix")
print(X_tr_set2.shape, y_train.shape)
print(X_cv_set2.shape, y_cv.shape)
print(X_te_set2.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 8874) (49041,)
(24155, 8874) (24155,)
(36052, 8874) (36052,)
====================================================================================================
```

In [131]:

```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Mo
from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression,SGDClassifier
import matplotlib.pyplot as plt

data = data #refer: http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.h

tuned_parameters = {'alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5]}

#Using SGDClassifier
model = GridSearchCV(SGDClassifier(loss='log', class_weight='balanced'), tuned_parameters, cv=5, scoring=
model.fit(X_tr_set2, y_train)

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

plt.figure()
plt.plot(tuned_parameters['alpha'],train_auc,label="Train AUC")
plt.gca().fill_between(tuned_parameters['alpha'],train_auc - train_auc_std,train_auc + train_auc_std,alpha

plt.plot(tuned_parameters['alpha'],cv_auc,label="CV AUC")
plt.gca().fill_between(tuned_parameters['alpha'],cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorang
plt.scatter(tuned_parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(tuned_parameters['alpha'], cv_auc, label='CV AUC points')

plt.legend(loc='best')
plt.xlabel("alpha:hyperparameters")
plt.ylabel("AUC")
plt.title("alpha:hyperparameters vs AUC Plot")
plt.show()
```



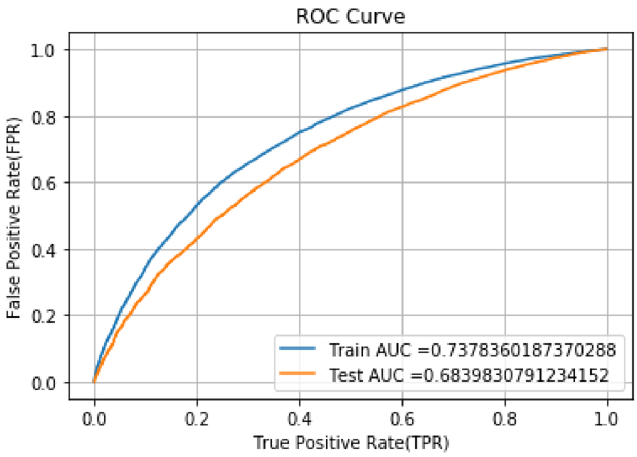
In [132]:

```
best_a = model.best_params_

best_a = list(best_a.values())[0]
print("Best a :{0}".format(best_a))
print(model.best_score_)
```

```
Best a :0.001
0.6861253937835214
```

In [133]:
```
def batch_predict(clf, data):
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class

# not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
# consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 49000
# in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
# we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [134]:
```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curv
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression

model = SGDClassifier(loss='log', alpha=best_a, class_weight='balanced')

model.fit(X_tr_set2, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr_set2)
y_test_pred = batch_predict(model, X_te_set2)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC Curve")
plt.grid(True)
plt.show()
```



In [135]:
```
# we are writing our own function for predict, with defined threshold
# 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.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [136]:

```
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_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_tpr)))
```

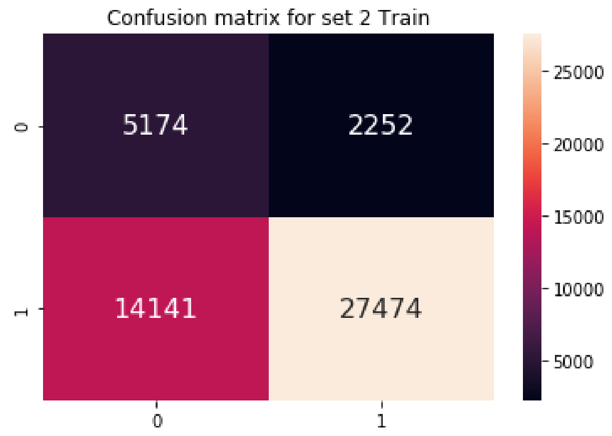
```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4599847932092946 for threshold 0.496
[[ 5174  2252]
 [14141 27474]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4599847932092946 for threshold 0.496
[[ 3385  2074]
 [10733 19860]]
```

In [205]:

```
#How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = [[5174,2252],
        [14141,27474]]
df_cm = pd.DataFrame(array, index = [i for i in "01"],
                  columns = [i for i in "01"])
plt.figure
plt.title('Confusion matrix for set 2 Train')
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

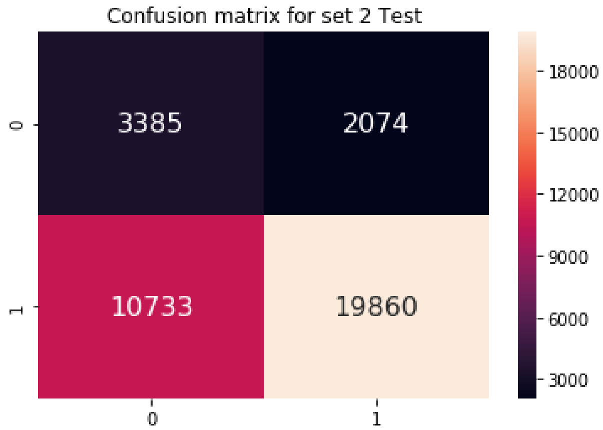
Out[205]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7c5443eb8>
```



```
In [206]: #How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
          import seaborn as sn
          import pandas as pd
          import matplotlib.pyplot as plt
          array = [[3385,2074],
                  [10733,19860]]
          df_cm = pd.DataFrame(array, index = [i for i in "01"],
                            columns = [i for i in "01"])
          plt.figure
          plt.title('Confusion matrix for set 2 Test')
          sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

```
Out[206]: <matplotlib.axes._subplots.AxesSubplot at 0x1e7c3ac2ac8>
```



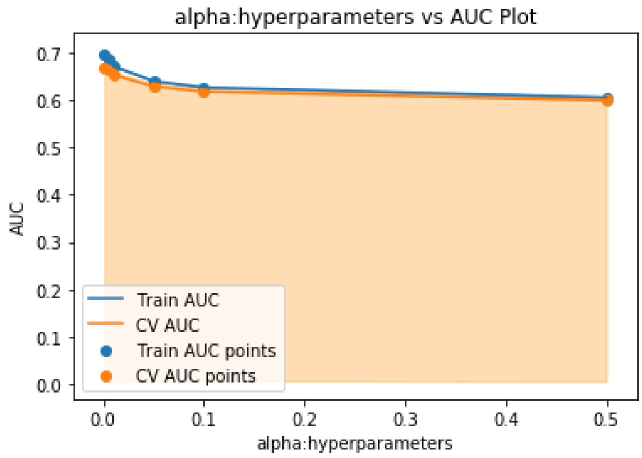
2.4.3 Applying Logistic Regression on AVG W2v, SET 3

Consider Set 3 :- categorical, numerical features + project_title(AVG W2V) + preprocessed_essay (AVG W2v)

```
In [139]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
          from scipy.sparse import hstack
          X_tr_set3 = hstack((avg_w2v_vectors_extr, avg_w2v_vectors_txtr, X_tr_numcat)).tocsr()
          X_cv_set3 = hstack((avg_w2v_vectors_excv, avg_w2v_vectors_txcv, X_cv_numcat)).tocsr()
          X_te_set3 = hstack((avg_w2v_vectors_exte, avg_w2v_vectors_txte, X_te_numcat )).tocsr()

          print("Final Data matrix")
          print(X_tr_set3.shape, y_train.shape)
          print(X_cv_set3.shape, y_cv.shape)
          print(X_te_set3.shape, y_test.shape)
          print("="*100)
```

```
Final Data matrix
(49041, 702) (49041,)
(24155, 702) (24155,)
(36052, 702) (36052,)
====================================================================================================
```

In [140]:
```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Mc
from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression,SGDClassifier
import matplotlib.pyplot as plt

data = data #refer: http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.h

tuned_parameters = {'alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5]}

#Using SGDClassifier
model = GridSearchCV(SGDClassifier(loss='log', class_weight='balanced'), tuned_parameters, cv=5, scoring=
model.fit(X_tr_set3, y_train)

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

plt.figure()
plt.plot(tuned_parameters['alpha'],train_auc,label="Train AUC")
plt.gca().fill_between(tuned_parameters['alpha'],train_auc - train_auc_std,train_auc + train_auc_std,alpha

plt.plot(tuned_parameters['alpha'],cv_auc,label="CV AUC")
plt.gca().fill_between(tuned_parameters['alpha'],cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorang
plt.scatter(tuned_parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(tuned_parameters['alpha'], cv_auc, label='CV AUC points')

plt.legend(loc='best')
plt.xlabel("alpha:hyperparameters")
plt.ylabel("AUC")
plt.title("alpha:hyperparameters vs AUC Plot")
plt.show()
```



In [141]:
```
best_a = model.best_params_

best_a = list(best_a.values())[0]
print("Best a :{0}".format(best_a))
print(model.best_score_)
```

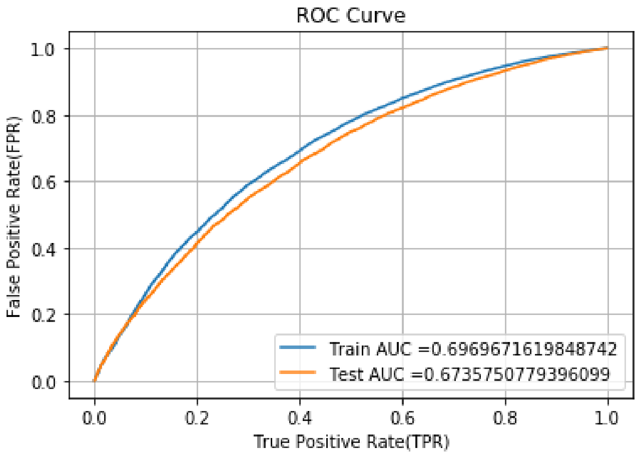
```
Best a :0.001
0.6675546667528117
```

In [142]:

```
def batch_predict(clf, data):
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class

# not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
# consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 49000
# in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
# we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [143]:

```

# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curv
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression

model = SGDClassifier(loss='log', alpha=best_a, class_weight='balanced')

model.fit(X_tr_set3, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr_set3)
y_test_pred = batch_predict(model, X_te_set3)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC Curve")
plt.grid(True)
plt.show()
```



In [144]:

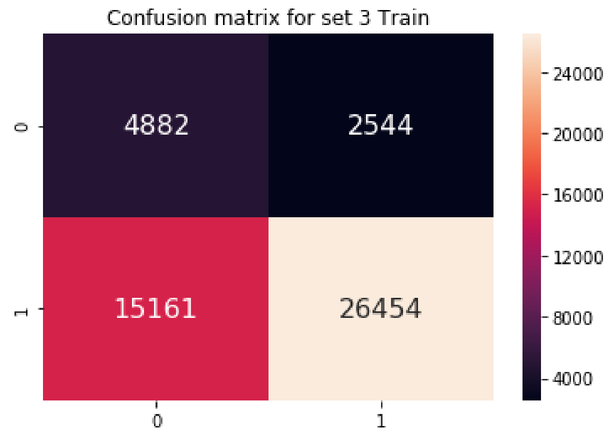
```
# we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    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.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [145]:
```
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_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_tpr)))
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4179114598735883 for threshold 0.386
[[ 4882  2544]
 [15161 26454]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4179114598735883 for threshold 0.386
[[ 3408  2051]
 [11448 19145]]
```

In [207]:
```
#How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = [[4882,2544],
        [15161,26454]]
df_cm = pd.DataFrame(array, index = [i for i in "01"],
                  columns = [i for i in "01"])
plt.figure
plt.title('Confusion matrix for set 3 Train')
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

Out[207]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7c2f8fb00>
```

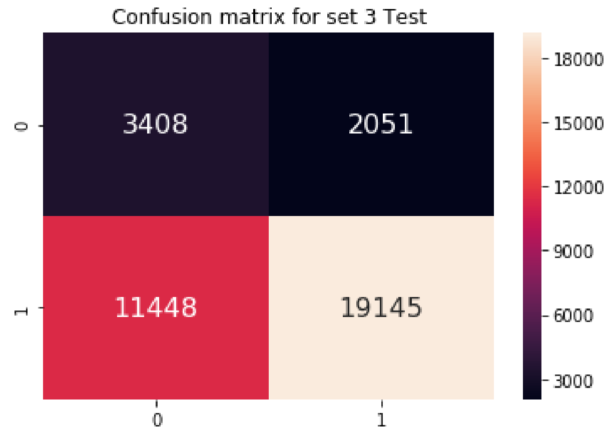


In [208]:

```
#How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = [[3408,2051],
        [11448,19145]]
df_cm = pd.DataFrame(array, index = [i for i in "01"],
                  columns = [i for i in "01"])
plt.figure
plt.title('Confusion matrix for set 3 Test')
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

Out[208]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7c3cc5160>
```



2.4.4 Applying Logistic Regression on TFIDF W2V, SET 4

Consider Set 4 :- categorical, numerical features + project_title(TFIDF w2v) + preprocessed_essay (TFIDF w2v)

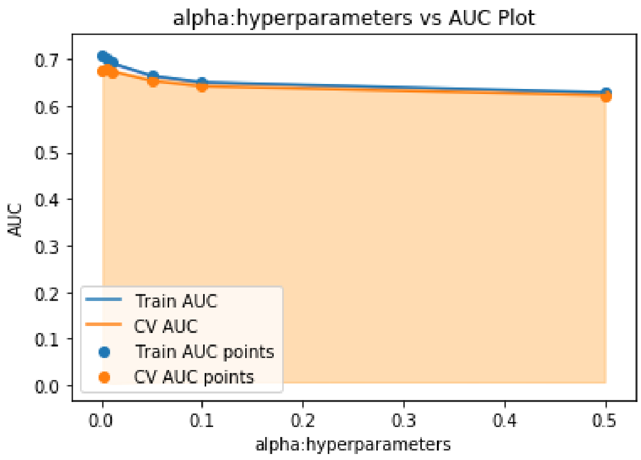
In [148]:

```
# Please write all the code with proper documentation
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_set4 = hstack((tfidf_w2v_vectors_extr, tfidf_w2v_vectors_txtr, X_tr_numcat)).tocsr()
X_cv_set4 = hstack((tfidf_w2v_vectors_excv, tfidf_w2v_vectors_txcv, X_cv_numcat)).tocsr()
X_te_set4 = hstack((tfidf_w2v_vectors_exte, tfidf_w2v_vectors_txte, X_te_numcat)).tocsr()

print("Final Data matrix")
print(X_tr_set4.shape, y_train.shape)
print(X_cv_set4.shape, y_cv.shape)
print(X_te_set4.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 702) (49041,)
(24155, 702) (24155,)
(36052, 702) (36052,)
====================================================================================================
```

In [149]:

```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Mo
from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression,SGDClassifier
import matplotlib.pyplot as plt

data = data #refer: http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.h

tuned_parameters = {'alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5]}

#Using SGDClassifier
model = GridSearchCV(SGDClassifier(loss='log', class_weight='balanced'), tuned_parameters, cv=5, scoring=
model.fit(X_tr_set4, y_train)

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

plt.figure()
plt.plot(tuned_parameters['alpha'],train_auc,label="Train AUC")
plt.gca().fill_between(tuned_parameters['alpha'],train_auc - train_auc_std,train_auc + train_auc_std,alpha

plt.plot(tuned_parameters['alpha'],cv_auc,label="CV AUC")
plt.gca().fill_between(tuned_parameters['alpha'],cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorang
plt.scatter(tuned_parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(tuned_parameters['alpha'], cv_auc, label='CV AUC points')

plt.legend(loc='best')
plt.xlabel("alpha:hyperparameters")
plt.ylabel("AUC")
plt.title("alpha:hyperparameters vs AUC Plot")
plt.show()
```



In [150]:

```
best_alpha = model.best_params_

best_alpha = list(best_alpha.values())[0]
print("Best alpha :{0}".format(best_alpha))
print(model.best_score_)
```

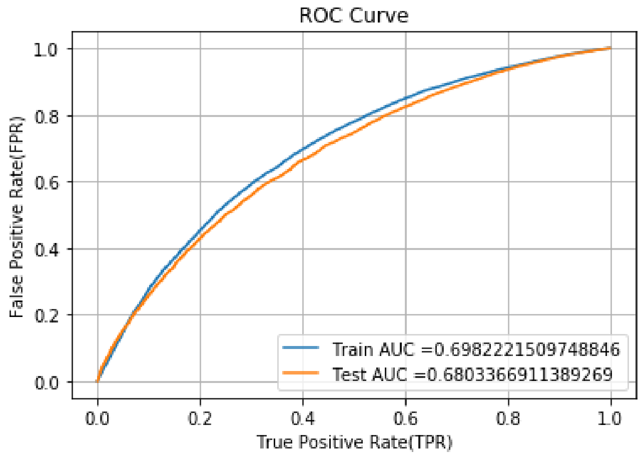
```
Best alpha :0.005
0.6794835126642572
```

In [151]:

```
def batch_predict(clf, data):

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
# consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 49000
# in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
# we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [152]:

```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curv
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression

model = SGDClassifier(loss='log', alpha=best_alpha, class_weight='balanced')

model.fit(X_tr_set4, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr_set4)
y_test_pred = batch_predict(model, X_te_set4)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC Curve")
plt.grid(True)
plt.show()
```



In [153]:

```
# we are writing our own function for predict, with defined threshold
# 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.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [154]:

```
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_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_tpr)))
```

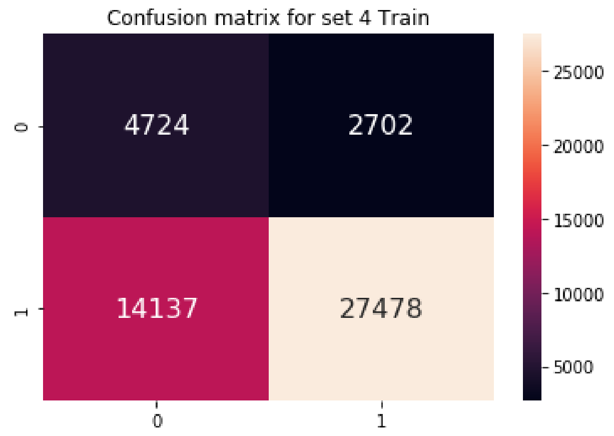
```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.42003953040741704 for threshold 0.483
[[ 4724  2702]
 [14137 27478]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.42003953040741704 for threshold 0.483
[[ 3351  2108]
 [10708 19885]]
```

In [209]:

```
#How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = [[4724,2702],
        [14137,27478]]
df_cm = pd.DataFrame(array, index = [i for i in "01"],
                  columns = [i for i in "01"])
plt.figure
plt.title('Confusion matrix for set 4 Train')
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

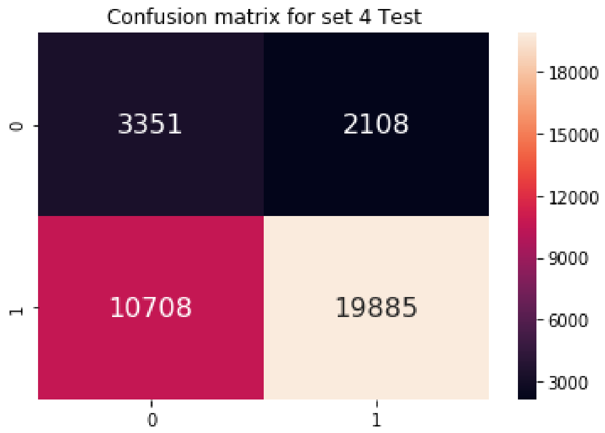
Out[209]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7c6f07d68>
```



```
In [210]: #How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
          import seaborn as sn
          import pandas as pd
          import matplotlib.pyplot as plt
          array = [[3351,2108],
                  [10708,19885]]
          df_cm = pd.DataFrame(array, index = [i for i in "01"],
                            columns = [i for i in "01"])
          plt.figure
          plt.title('Confusion matrix for set 4 Test')
          sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

```
Out[210]: <matplotlib.axes._subplots.AxesSubplot at 0x1e7c50b6320>
```



2.5 Logistic Regression with added Features `Set 5`

```
In [ ]: # please write all the code with proper documentation, and proper titles for each subsection
        # go through documentations and blogs before you start coding
        # first figure out what to do, and then think about how to do.
        # reading and understanding error messages will be very much helpfull in debugging your code
        # when you plot any graph make sure you use
            # a. Title, that describes your plot, this will be very helpful to the reader
            # b. Legends if needed
            # c. X-axis label
            # d. Y-axis label
```

Counting the number of words in Essays

```
In [157]: project_data["preprocessed_essays"] = preprocessed_essays
```

```
In [158]: essay_words_total = []
          for _ in project_data["preprocessed_essays"]:
              x = len(_.split())
              essay_words_total.append(x)
```

```
In [159]: project_data["essay_words_total"] = essay_words_total
```

In [160]:
```
project_data.head()
```

Out[160]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs.	in	2016-12-05 13:43:57	grade
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr.	fl	2016-10-25 09:22:10	gr
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	ms.	az	2016-08-31 12:03:56	gr
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	mrs.	ky	2016-10-06 21:16:17	grade
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	mrs.	tx	2016-07-11 01:10:09	grade

5 rows × 21 columns

In [166]:
```
project_data.drop(['essay'], axis=1, inplace=True)
```

In [168]:
```
project_data.head()
```

Out[168]:

eviously_posted_projects	clean_categories	clean_subcategories	price	quantity	preprocessed_essays	essay_words_total	preproc
0	Literacy_Language	ESL Literacy	154.60	23	my students english learners working english s...	160	educat eng
7	History_Civics Health_Sports	Civics_Government TeamSports	299.00	1	our students arrive school eager learn they po...	108	wan hu
1	Health_Sports	Health_Wellness TeamSports	516.85	22	true champions not always ones win guts by mia...	201	socc awe sch
4	Literacy_Language Math_Science	Literacy Mathematics	232.90	4	i work unique school filled esl english second...	120	kin
1	Math_Science	Mathematics	67.98	4	our second grade classroom next year made arou...	121	interactiv

Counting the number of words in Project Title

In [161]:
```
project_data["preprocessed_titles"] = preprocessed_titles
```

In [162]:
```
project_data.head()
```

Out[162]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs.	in	2016-12-05 13:43:57	grade
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr.	fl	2016-10-25 09:22:10	gr
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	ms.	az	2016-08-31 12:03:56	gr
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	mrs.	ky	2016-10-06 21:16:17	grade
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	mrs.	tx	2016-07-11 01:10:09	grade

5 rows × 22 columns

In [163]:
```
title_words_total = []
for _ in project_data["preprocessed_titles"]:
    y = len(_.split())
    title_words_total.append(y)
```

In [164]:
```
project_data["title_words_total"] = title_words_total
```

In [165]:
```
project_data.head()
```

Out[165]:

sted_projects	clean_categories	clean_subcategories	essay	price	quantity	preprocessed_essays	essay_words_total	preproc
0	Literacy_Language	ESL Literacy	My students are English learners that are work...	154.60	23	my students english learners working english s...	160	educat eng
7	History_Civics Health_Sports	Civics_Government TeamSports	Our students arrive to our school eager to lea...	299.00	1	our students arrive school eager learn they po...	108	war hu
1	Health_Sports	Health_Wellness TeamSports	\r\n"True champions aren't always the ones th...	516.85	22	true champions not always ones win guts by mia...	201	socc awe scl
4	Literacy_Language Math_Science	Literacy Mathematics	I work at a unique school filled with both ESL...	232.90	4	i work unique school filled esl english second...	120	kin
1	Math_Science	Mathematics	Our second grade classroom next year will be m...	67.98	4	our second grade classroom next year made arou...	121	interactiv

Calculate sentiment score for essays

In [169]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()
neg = []
pos = []
neu = []
compound = []

for _ in tqdm(project_data["preprocessed_essays"]) :
    w = sid.polarity_scores(_)['neg']
    x = sid.polarity_scores(_)['pos']
    y = sid.polarity_scores(_)['neu']
    z = sid.polarity_scores(_)['compound']
    neg.append(w)
    pos.append(x)
    neu.append(y)
    compound.append(z)
```

```
100%|██████████████████████████████████████████████████████████████████████| 109248/109248 [14:40<00:
00, 124.11it/s]
```

In [170]:
```
project_data["pos"] = pos
```

In [171]:
```
project_data["neg"] = neg
```

In [172]:
```
project_data["neu"] = neu
```

In [173]:
```
project_data["compound"] = compound
```

In [174]:

```python
project_data.head()
```

Out[174]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs.	in	2016-12-05 13:43:57	grade
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	mr.	fl	2016-10-25 09:22:10	gr
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	ms.	az	2016-08-31 12:03:56	gr
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	mrs.	ky	2016-10-06 21:16:17	grade
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	mrs.	tx	2016-07-11 01:10:09	grade

5 rows × 26 columns

In [176]:

```python
data5 = project_data
```

In [177]:

```python
x5 = data5
```

Splitting data into Train and cross validation(or test)

In [178]:

```python
#Splitting data into test & train set
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
from sklearn.model_selection import train_test_split
X5_train, X5_test = train_test_split(X,test_size = 0.33)
```

In [179]:

```python
#Splitting training data into training & cross validation sets
X5_train, X5_cv = train_test_split(X5_train,test_size = 0.33)
```

In [180]:

```python
print(X5_train.shape)
print(X5_cv.shape)
print(X5_test.shape)
```

```
(49041, 26)
(24155, 26)
(36052, 26)
```

Normalising Essay word count

In [183]:

```
from sklearn.preprocessing import Normalizer
essay_words_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
essay_words_norm.fit(X5_train['essay_words_total'].values.reshape(1,-1))

X5_train_ewords_norm = essay_words_norm.transform(X5_train['essay_words_total'].values.reshape(1,-1))
X5_cv_ewords_norm = essay_words_norm.transform(X5_cv['essay_words_total'].values.reshape(1,-1))
X5_test_ewords_norm = essay_words_norm.transform(X5_test['essay_words_total'].values.reshape(1,-1))

print("After vectorizations")
print(X5_train_ewords_norm.shape, y_train.shape)
print(X5_cv_ewords_norm.shape, y_cv.shape)
print(X5_test_ewords_norm.shape, y_test.shape)
print("="*100)
```

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

In [184]:

```
X5_train_ewords_norm = X5_train_ewords_norm.T
X5_cv_ewords_norm = X5_cv_ewords_norm.T
X5_test_ewords_norm = X5_test_ewords_norm.T

print("Final Matrix")
print(X5_train_ewords_norm.shape, y_train.shape)
print(X5_cv_ewords_norm.shape, y_cv.shape)
print(X5_test_ewords_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

Normalising Project Title word count

In [185]:

```
from sklearn.preprocessing import Normalizer
title_words_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
title_words_norm.fit(X5_train['title_words_total'].values.reshape(1,-1))

X5_train_twords_norm = title_words_norm.transform(X5_train['title_words_total'].values.reshape(1,-1))
X5_cv_twords_norm = title_words_norm.transform(X5_cv['title_words_total'].values.reshape(1,-1))
X5_test_twords_norm = title_words_norm.transform(X5_test['title_words_total'].values.reshape(1,-1))

print("After vectorizations")
print(X5_train_twords_norm.shape, y_train.shape)
print(X5_cv_twords_norm.shape, y_cv.shape)
print(X5_test_twords_norm.shape, y_test.shape)
print("="*100)
```

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

In [186]:

```
X5_train_twords_norm = X5_train_twords_norm.T
X5_cv_twords_norm = X5_cv_twords_norm.T
X5_test_twords_norm = X5_test_twords_norm.T

print("Final Matrix")
print(X5_train_twords_norm.shape, y_train.shape)
print(X5_cv_twords_norm.shape, y_cv.shape)
print(X5_test_twords_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

Normalising Essay Sentiment scores

Normalising positive score

In [187]:

```
from sklearn.preprocessing import Normalizer
senti_pos_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
senti_pos_norm.fit(X5_train['pos'].values.reshape(1,-1))

X5_train_pos_norm = senti_pos_norm.transform(X5_train['pos'].values.reshape(1,-1))
X5_cv_pos_norm = senti_pos_norm.transform(X5_cv['pos'].values.reshape(1,-1))
X5_test_pos_norm = senti_pos_norm.transform(X5_test['pos'].values.reshape(1,-1))

print("After vectorizations")
print(X5_train_pos_norm.shape, y_train.shape)
print(X5_cv_pos_norm.shape, y_cv.shape)
print(X5_test_pos_norm.shape, y_test.shape)
print("="*100)
```

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

In [188]:

```
X5_train_pos_norm = X5_train_pos_norm.T
X5_cv_pos_norm = X5_cv_pos_norm.T
X5_test_pos_norm = X5_test_pos_norm.T

print("Final Matrix")
print(X5_train_pos_norm.shape, y_train.shape)
print(X5_cv_pos_norm.shape, y_cv.shape)
print(X5_test_pos_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

Normalising Negative score

In [189]:

```
from sklearn.preprocessing import Normalizer
senti_neg_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
senti_neg_norm.fit(X5_train['neg'].values.reshape(1,-1))

X5_train_neg_norm = senti_neg_norm.transform(X5_train['neg'].values.reshape(1,-1))
X5_cv_neg_norm = senti_neg_norm.transform(X5_cv['neg'].values.reshape(1,-1))
X5_test_neg_norm = senti_neg_norm.transform(X5_test['neg'].values.reshape(1,-1))

print("After vectorizations")
print(X5_train_neg_norm.shape, y_train.shape)
print(X5_cv_neg_norm.shape, y_cv.shape)
print(X5_test_neg_norm.shape, y_test.shape)
print("="*100)
```

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

In [190]:

```
X5_train_neg_norm = X5_train_neg_norm.T
X5_cv_neg_norm = X5_cv_neg_norm.T
X5_test_neg_norm = X5_test_neg_norm.T

print("Final Matrix")
print(X5_train_neg_norm.shape, y_train.shape)
print(X5_cv_neg_norm.shape, y_cv.shape)
print(X5_test_neg_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

Normalising Neutral scores

In [191]:

```
from sklearn.preprocessing import Normalizer
senti_neu_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
senti_neu_norm.fit(X5_train['neu'].values.reshape(1,-1))

X5_train_neu_norm = senti_neu_norm.transform(X5_train['neu'].values.reshape(1,-1))
X5_cv_neu_norm = senti_neu_norm.transform(X5_cv['neu'].values.reshape(1,-1))
X5_test_neu_norm = senti_neu_norm.transform(X5_test['neu'].values.reshape(1,-1))

print("After vectorizations")
print(X5_train_neu_norm.shape, y_train.shape)
print(X5_cv_neu_norm.shape, y_cv.shape)
print(X5_test_neu_norm.shape, y_test.shape)
print("="*100)
```

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

In [192]:

```
X5_train_neu_norm = X5_train_neu_norm.T
X5_cv_neu_norm = X5_cv_neu_norm.T
X5_test_neu_norm = X5_test_neu_norm.T

print("Final Matrix")
print(X5_train_neu_norm.shape, y_train.shape)
print(X5_cv_neu_norm.shape, y_cv.shape)
print(X5_test_neu_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

Normalising Compound scores

In [193]:

```
from sklearn.preprocessing import Normalizer
senti_comp_norm = Normalizer()
# normalizer.fit(X5_train['essay_word_total'].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.
senti_comp_norm.fit(X5_train['compound'].values.reshape(1,-1))

X5_train_comp_norm = senti_comp_norm.transform(X5_train['compound'].values.reshape(1,-1))
X5_cv_comp_norm = senti_comp_norm.transform(X5_cv['compound'].values.reshape(1,-1))
X5_test_comp_norm = senti_comp_norm.transform(X5_test['compound'].values.reshape(1,-1))

print("After vectorizations")
print(X5_train_comp_norm.shape, y_train.shape)
print(X5_cv_comp_norm.shape, y_cv.shape)
print(X5_test_comp_norm.shape, y_test.shape)
print("="*100)
```

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

In [194]:

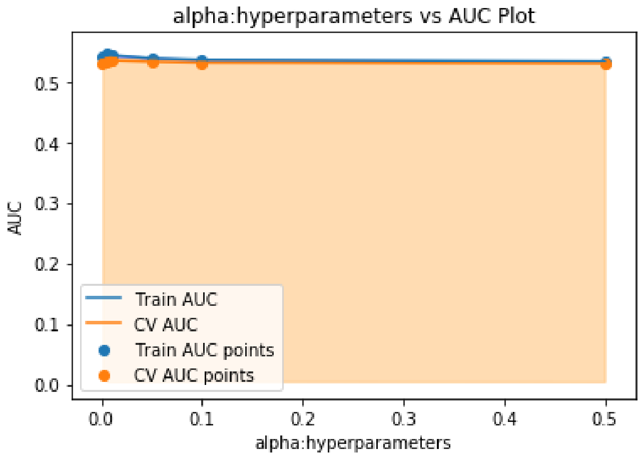
```
X5_train_comp_norm = X5_train_comp_norm.T
X5_cv_comp_norm = X5_cv_comp_norm.T
X5_test_comp_norm = X5_test_comp_norm.T

print("Final Matrix")
print(X5_train_comp_norm.shape, y_train.shape)
print(X5_cv_comp_norm.shape, y_cv.shape)
print(X5_test_comp_norm.shape, y_test.shape)
print("="*100)
```

```
Final Matrix
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================================
```

In [195]:
```
# Please write all the code with proper documentation
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr_set5 = hstack((X5_train_pos_norm, X5_train_neg_norm, X5_train_neu_norm, X5_train_comp_norm, X5_train_
X_cv_set5 = hstack((X5_cv_pos_norm, X5_cv_neg_norm, X5_cv_neu_norm, X5_cv_comp_norm, X5_cv_ewords_norm, X5
X_te_set5 = hstack((X5_test_pos_norm, X5_test_neg_norm, X5_test_neu_norm, X5_test_comp_norm, X5_test_eword

print("Final Data matrix")
print(X_tr_set5.shape, y_train.shape)
print(X_cv_set5.shape, y_cv.shape)
print(X_te_set5.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 108) (49041,)
(24155, 108) (24155,)
(36052, 108) (36052,)
====================================================================================================
```

In [196]:
```
#code source: http://occam.olin.edu/sites/default/files/DataScienceMaterials/machine_learning_lecture_2/Mo
from sklearn.model_selection import learning_curve, GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression,SGDClassifier
import matplotlib.pyplot as plt

data = data #refer: http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.h

tuned_parameters = {'alpha': [0.001, 0.005, 0.01, 0.05, 0.1, 0.5]}

#Using SGDClassifier
model = GridSearchCV(SGDClassifier(loss='log', class_weight='balanced'), tuned_parameters, cv=5, scoring=
model.fit(X_tr_set5, y_train)

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

plt.figure()
plt.plot(tuned_parameters['alpha'],train_auc,label="Train AUC")
plt.gca().fill_between(tuned_parameters['alpha'],train_auc - train_auc_std,train_auc + train_auc_std,alpha

plt.plot(tuned_parameters['alpha'],cv_auc,label="CV AUC")
plt.gca().fill_between(tuned_parameters['alpha'],cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkoran
plt.scatter(tuned_parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(tuned_parameters['alpha'], cv_auc, label='CV AUC points')

plt.legend(loc='best')
plt.xlabel("alpha:hyperparameters")
plt.ylabel("AUC")
plt.title("alpha:hyperparameters vs AUC Plot")
plt.show()
```



In [197]:

```
best_alpha = model.best_params_

best_alpha = list(best_alpha.values())[0]
print("Best alpha :{0}".format(best_alpha))
print(model.best_score_)
```

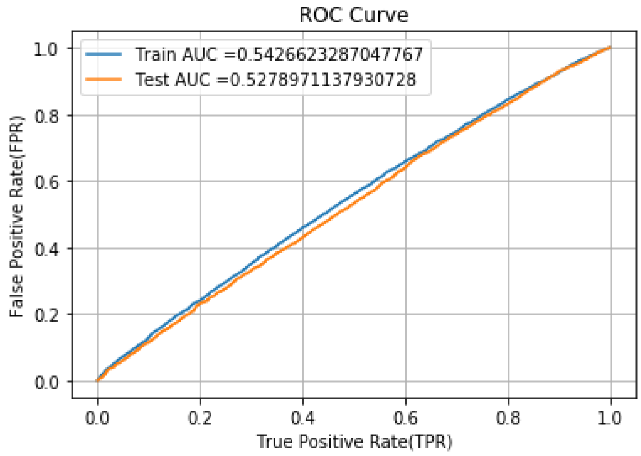
```
Best alpha :0.01
0.5350017350658864
```

In [198]:

```
def batch_predict(clf, data):

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
# consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 49000
# in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
# we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [199]:

```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curv
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression

model = SGDClassifier(loss='log', alpha=best_alpha, class_weight='balanced')

model.fit(X_tr_set5, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr_set5)
y_test_pred = batch_predict(model, X_te_set5)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC Curve")
plt.grid(True)
plt.show()
```



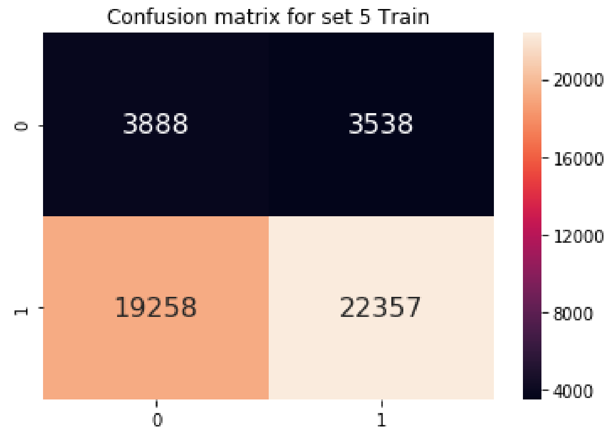
In [200]:
```
# we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    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.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [201]:
```
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_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, train_fpr, train_tpr)))
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2812774649075491 for threshold 0.507
[[ 3888  3538]
 [19258 22357]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2812774649075491 for threshold 0.507
[[ 2746  2713]
 [14409 16184]]
```

In [211]:
```
#How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = [[3888,3538],
        [19258,22357]]
df_cm = pd.DataFrame(array, index = [i for i in "01"],
                  columns = [i for i in "01"])
plt.figure
plt.title('Confusion matrix for set 5 Train')
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

Out[211]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7c1150be0>
```

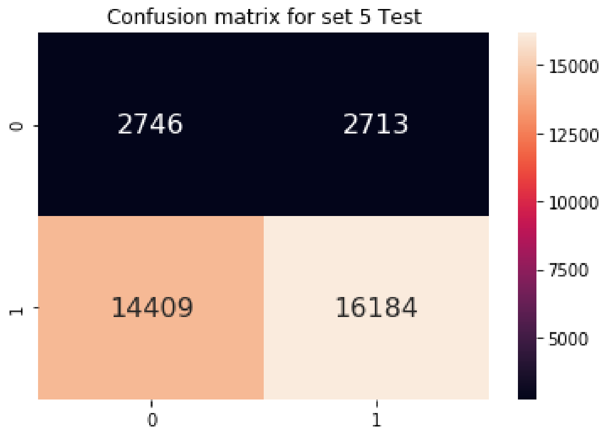


In [212]:

```
#How to plot confusion matrix using heat map - https://stackoverflow.com/questions/35572000/how-can-i-plot
import seaborn as sn
import pandas as pd
import matplotlib.pyplot as plt
array = [[2746,2713],
        [14409,16184]]
df_cm = pd.DataFrame(array, index = [i for i in "01"],
                  columns = [i for i in "01"])
plt.figure
plt.title('Confusion matrix for set 5 Test')
sn.heatmap(df_cm, annot=True, annot_kws={"size":16}, fmt='g')
```

Out[212]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e7c7cb9f28>
```



3. Conclusion

In [202]:

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

from prettytable import PrettyTable

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

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Alpha:Hyper Parameter", "AUC"]

x.add_row(["BOW", "Logistic Regression", 0.01, 0.70])
x.add_row(["TFIDF", "Logistic Regression", 0.001, 0.68])
x.add_row(["AVG W2V", "Logistic Regression", 0.001, 0.67])
x.add_row(["TFIDF W2V", "Logistic Regression", 0.005, 0.68])
x.add_row(["WITHOUT TEXT DATA", "Logistic Regression", 0.01, 0.52])

print(x)
```

```
+-------------------+---------------------+-----------------------+------+
|     Vectorizer    |        Model        | Alpha:Hyper Parameter | AUC  |
+-------------------+---------------------+-----------------------+------+
|        BOW        | Logistic Regression |          0.01         | 0.7  |
|       TFIDF       | Logistic Regression |         0.001         | 0.68 |
|      AVG W2V      | Logistic Regression |         0.001         | 0.67 |
|     TFIDF W2V     | Logistic Regression |         0.005         | 0.68 |
| WITHOUT TEXT DATA | Logistic Regression |          0.01         | 0.52 |
+-------------------+---------------------+-----------------------+------+
```