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`

Feature	Description
project_subject_categories	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • Applied Learning • Care & Hunger • Health & Sports • History & Civics • Literacy & Language • Math & Science • Music & The Arts • Special Needs • Warmth **Examples:** • Music & The Arts • Literacy & Language, Math & Science
school_state	State where school is located (Two-letter U.S. postal code (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **Example:** WY
project_subject_subcategories	One or more (comma-separated) subject subcategories for the project. **Examples:** • Literacy • Literature & Writing, Social Sciences
project_resource_summary	An explanation of the resources needed for the project. **Example:** • My students need hands on literacy materials to manage sensory needs! </code
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Example:** 2016-04-28 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed project. **Example:** bdf8baa8fedef6bfeec7ae4ff1c15c56

Feature	Description
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', nrows=50000)
resource_data = pd.read_csv('resources.csv')
```

In [3]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

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

In [4]:
```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


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


project_data.head(2)
```

Out[4]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_subject_categc
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Grades PreK-2	Applied Lear
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Grades 3-5	Literacy & Langı

In [5]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[5]:

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

1.2 preprocessing of `project_subject_categories`

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removi
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&
        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 [7]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

In [8]:
```
# 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 [9]:
```
project_data.head(2)
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_title	project_e
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	Grades PreK-2	Flexible Seating for Flexible Learning	I recently artic giving s
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Grades 3-5	Going Deep: The Art of Inner Thinking!	My s crave ch ob

In [10]:
```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [11]:
```
# 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)
```

```
I recently read an article about giving students a choice about how they learn. We already set goals; why no
t let them choose where to sit, and give them options of what to sit on?I teach at a low-income (Title 1) sc
hool. Every year, I have a class with a range of abilities, yet they are all the same age. They learn differ
ently, and they have different interests. Some have ADHD, and some are fast learners. Yet they are eager and
active learners that want and need to be able to move around the room, yet have a place that they can be com
fortable to complete their work.We need a classroom rug that we can use as a class for reading time, and stu
dents can use during other learning times. I have also requested four Kore Kids wobble chairs and four Back
Jack padded portable chairs so that students can still move during whole group lessons without disrupting th
e class. Having these areas will provide these little ones with a way to wiggle while working.Benjamin Frank
lin once said, \"Tell me and I forget, teach me and I may remember, involve me and I learn.\" I want these c
hildren to be involved in their learning by having a choice on where to sit and how to learn, all by giving
them options for comfortable flexible seating.
==================================================
At the beginning of every class we start out with a Math Application problem to help students see the releva
nce of topics in math. We are always in groups and do a lot of cooperative activities. We also use lots of t
echnology in our class. I love seeing my students grow and love math!I have a very diverse population of stu
dents from all different races, SES, and experiences. My students love school and are starting to embrace th
e hard work it takes to be a fifth grader.  My school is a 5th/6th grade school only and is considered a sch
ool for the middle grades. It is located in a suburban area. It is now more diverse than it has been in many
years.I am in an inclusion setting and many of my students have disabilities. It is hard for them to see the
board because our resources are old and outdated.  A new document camera for our classroom will allow our st
udents to see the board more clearly during instructional times and will create a classroom environment wher
e lots of movement isn't necessary just because my students cannot see the board.It's frustrating to teach a
lesson when many of my students can't see the board because the resources I have are old and outdated. Often
times students will tell me to wait before moving on because it takes them forever to write notes because th
ey cannot see the materials.  I want students to enjoy coming to my class to learn math and not feel frustra
ted because they cannot see the board.
==================================================
My students love coming to school and they love learning. I strive daily to make our classroom a relaxed, co
mfortable and welcoming environment where all learners will excel and grow in their learning. And a new rug
will make our days even brighter!My 2nd grade classroom is filled with 20 amazing young learners. These stud
```

ents fill my heart everyday with their passion for learning new things. Working with these students and how engaged they are in each subject matter is so much fun. We are small elementary school in mid-Missouri and we have an 80 percent free and reduced lunch rate. I have a wide range of learners in my classroom, and all of my students learn in different ways. So it is important to provide a learning environment that meets all students.A beautiful new carpet will be the focal point of our classroom. The carpet will be full of students all day long. It will be a clean and comfortable place where my students will find comfort in learning. Students will be sitting in small groups, laying and reading a book or even dancing on the carpet for brain breaks during the day. A carpet in an elementary classroom is the heart of where learning takes place!Thank you for donating or considering a donation to this project. I want to make my 2nd grade classroom as comfortable and inviting as Starbucks or as cozy as a grandma's living room! This beautiful carpet will be a perfect addition to a classroom the is filled with so much excitement and enthusiasm!

==========

I teach at a Title 1 school, with 73% of my students who receive free/reduced lunch. Our school provides free breakfast for all students. I am a Special Education certified teacher and I teach Kindergarten in a general education setting with my class that consists 52% students with special needs. The disabilities include Autism Spectrum Disorder, Speech Impaired, Language Impaired, Other Health Impaired (ADHD), and Developmentally Delayed. I also have about 42% of my students who are English Language Learners.\r\n\r\n\"Self-motivated learners\" is a synonym of \"my students\". They love to learn and they possess a positive outlook and attitude in school. Almost everyday, my students would ask me, \"Ms. Perez, what are we going to learn today?\" I could not ask for a better greeting from my students.This project will greatly impact my students' learning on a daily basis. The wobble chairs will provide assistance for my students who have difficulties focusing and attending during lessons and discussions. Despite the fact that students participate in physical activities in P.E., Recess, and GoNoodle (dance videos) sessions in our classroom, students still have energy to stand or wiggle from their seats during lessons. Due to these special needs that are beyond the students' control, there is a lot of distraction and student learning is not really achieved at its full potential. The lack of appropriate stimulation hinders them to focus and learn in class. Students with special needs will be able to sit on the wobble chairs during whole group/small group lessons. This will enable their little active bodies to move while "sitting still" without disrupting other students. As a result, all students will improve focus and increase student attention in learning all content areas. In addition, the visual timer will help my students to actually see the allotted time for activities. This will benefit especially ELL students and students with special needs. Whenever we do independent classwork or work in our centers, the students can refer to it and self-monitor their progress in completing assignments. It will encourage them to use their time wisely and finish tasks on time. It will also help the students have a smoother transition from one activity to another. \r\nBy donating to this project, you will significantly help students with special needs have an equal opportunity to learn with their peers. Behavior issues will be greatly minimized and classroom management will be optimized. Help me set all students for success! I am looking forward to seeing my students become active listeners and engaged learners, and always happy to go to school!\r\nnannan

==========

In [12]:
```
# 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 [13]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
I teach at a Title 1 school, with 73% of my students who receive free/reduced lunch. Our school provides fre
e breakfast for all students. I am a Special Education certified teacher and I teach Kindergarten in a gener
al education setting with my class that consists 52% students with special needs. The disabilities include A
utism Spectrum Disorder, Speech Impaired, Language Impaired, Other Health Impaired (ADHD), and Developmental
ly Delayed. I also have about 42% of my students who are English Language Learners.\r\n\r\n\"Self-motivated
 learners\" is a synonym of \"my students\".  They love to learn and they possess a positive outlook and atti
tude in school. Almost everyday, my students would ask me, \"Ms. Perez, what are we going to learn today?\"
 I could not ask for a better greeting from my students.This project will greatly impact my students' learnin
g on a daily basis. The wobble chairs will provide assistance for my students who have difficulties focusing
 and attending during lessons and discussions. Despite the fact that students participate in physical activit
ies in P.E., Recess, and GoNoodle (dance videos) sessions in our classroom, students still have energy to st
and or wiggle from their seats during lessons. Due to these special needs that are beyond the students' cont
rol, there is a lot of distraction and student learning is not really achieved at its full potential. The la
ck of appropriate stimulation hinders them to focus and learn in class. Students with special needs will be
 able to sit on the wobble chairs during whole group/small group lessons. This will enable their little activ
e bodies to move while "sitting still" without disrupting other students. As a result, all students will imp
rove focus and increase student attention in learning all content areas. In addition, the visual timer will
 help my students to actually see the allotted time for activities. This will benefit especially ELL students
 and students with special needs. Whenever we do independent classwork or work in our centers, the students c
an refer to it and self-monitor their progress in completing assignments. It will encourage them to use thei
r time wisely and finish tasks on time. It will also help the students have a smoother transition from one a
ctivity to another. \r\nBy donating to this project, you will significantly help students with special needs
 have an equal opportunity to learn with their peers. Behavior issues will be greatly minimized and classroom
 management will be optimized. Help me set all students for success! I am looking forward to seeing my studen
ts become active listeners and engaged learners, and always happy to go to school!\r\nnannan
==================================================
```

In [14]:
```
# \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)
```

```
I teach at a Title 1 school, with 73% of my students who receive free/reduced lunch. Our school provides fre
e breakfast for all students. I am a Special Education certified teacher and I teach Kindergarten in a gener
al education setting with my class that consists 52% students with special needs. The disabilities include A
utism Spectrum Disorder, Speech Impaired, Language Impaired, Other Health Impaired (ADHD), and Developmental
ly Delayed. I also have about 42% of my students who are English Language Learners.     Self-motivated learn
ers  is a synonym of  my students .  They love to learn and they possess a positive outlook and attitude in
school. Almost everyday, my students would ask me,  Ms. Perez, what are we going to learn today?  I could no
t ask for a better greeting from my students.This project will greatly impact my students' learning on a dai
ly basis. The wobble chairs will provide assistance for my students who have difficulties focusing and atten
ding during lessons and discussions. Despite the fact that students participate in physical activities in P.
E., Recess, and GoNoodle (dance videos) sessions in our classroom, students still have energy to stand or wi
ggle from their seats during lessons. Due to these special needs that are beyond the students' control, ther
e is a lot of distraction and student learning is not really achieved at its full potential. The lack of app
ropriate stimulation hinders them to focus and learn in class. Students with special needs will be able to s
it on the wobble chairs during whole group/small group lessons. This will enable their little active bodies
to move while "sitting still" without disrupting other students. As a result, all students will improve focu
s and increase student attention in learning all content areas. In addition, the visual timer will help my s
tudents to actually see the allotted time for activities. This will benefit especially ELL students and stud
ents with special needs. Whenever we do independent classwork or work in our centers, the students can refer
to it and self-monitor their progress in completing assignments. It will encourage them to use their time wi
sely and finish tasks on time. It will also help the students have a smoother transition from one activity t
o another.   By donating to this project, you will significantly help students with special needs have an eq
ual opportunity to learn with their peers. Behavior issues will be greatly minimized and classroom managemen
t will be optimized. Help me set all students for success! I am looking forward to seeing my students become
active listeners and engaged learners, and always happy to go to school!  nannan
```

In [15]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
I teach at a Title 1 school with 73 of my students who receive free reduced lunch Our school provides free b
reakfast for all students I am a Special Education certified teacher and I teach Kindergarten in a general e
ducation setting with my class that consists 52 students with special needs The disabilities include Autism
Spectrum Disorder Speech Impaired Language Impaired Other Health Impaired ADHD and Developmentally Delayed I
also have about 42 of my students who are English Language Learners Self motivated learners is a synonym of
my students They love to learn and they possess a positive outlook and attitude in school Almost everyday my
students would ask me Ms Perez what are we going to learn today I could not ask for a better greeting from m
y students This project will greatly impact my students learning on a daily basis The wobble chairs will pro
vide assistance for my students who have difficulties focusing and attending during lessons and discussions
Despite the fact that students participate in physical activities in P E Recess and GoNoodle dance videos se
ssions in our classroom students still have energy to stand or wiggle from their seats during lessons Due to
these special needs that are beyond the students control there is a lot of distraction and student learning
is not really achieved at its full potential The lack of appropriate stimulation hinders them to focus and l
earn in class Students with special needs will be able to sit on the wobble chairs during whole group small
group lessons This will enable their little active bodies to move while sitting still without disrupting oth
er students As a result all students will improve focus and increase student attention in learning all conte
nt areas In addition the visual timer will help my students to actually see the allotted time for activities
This will benefit especially ELL students and students with special needs Whenever we do independent classwo
rk or work in our centers the students can refer to it and self monitor their progress in completing assignm
ents It will encourage them to use their time wisely and finish tasks on time It will also help the students
have a smoother transition from one activity to another By donating to this project you will significantly h
elp students with special needs have an equal opportunity to learn with their peers Behavior issues will be
greatly minimized and classroom management will be optimized Help me set all students for success I am looki
ng forward to seeing my students become active listeners and engaged learners and always happy to go to scho
ol nannan
```

In [16]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'thos
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'd
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'd
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'fe
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o
            '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", 'must
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren
            'won', "won't", 'wouldn', "wouldn't"]
```

In [17]:
```
# 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.lower() not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 50000/50000 [00:47<00:00, 1054.58it/s]
```

In [18]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[18]:
```
'teach title 1 school 73 students receive free reduced lunch school provides free breakfast students special
education certified teacher teach kindergarten general education setting class consists 52 students special
needs disabilities include autism spectrum disorder speech impaired language impaired health impaired adhd d
evelopmentally delayed also 42 students english language learners self motivated learners synonym students l
ove learn possess positive outlook attitude school almost everyday students would ask ms perez going learn t
oday could not ask better greeting students project greatly impact students learning daily basis wobble chai
rs provide assistance students difficulties focusing attending lessons discussions despite fact students par
ticipate physical activities p e recess gonoodle dance videos sessions classroom students still energy stand
wiggle seats lessons due special needs beyond students control lot distraction student learning not really a
chieved full potential lack appropriate stimulation hinders focus learn class students special needs able si
t wobble chairs whole group small group lessons enable little active bodies move sitting still without disru
pting students result students improve focus increase student attention learning content areas addition visu
al timer help students actually see allotted time activities benefit especially ell students students specia
l needs whenever independent classwork work centers students refer self monitor progress completing assignme
nts encourage use time wisely finish tasks time also help students smoother transition one activity another
donating project significantly help students special needs equal opportunity learn peers behavior issues gre
atly minimized classroom management optimized help set students success looking forward seeing students beco
me active listeners engaged learners always happy go school nannan'
```

1.4 Preprocessing of `project_title`

In [19]:
```
preprocessed_titles = []
for titles in tqdm(project_data["project_title"].values):
    title = decontracted(titles)
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = title.replace('--', ' ')
    title = ' '.join(e for e in title.split() if e not in stopwords)

    preprocessed_titles.append(title.lower().strip())
```

```
100%|██████████| 50000/50000 [00:02<00:00, 24079.07it/s]
```

1.4.1 Preprocessing project_grade_categories

In [20]:
```
preprocessed_grades = []
for grade in tqdm(project_data["project_grade_category"].values):

    title = re.sub(' ', '', grade)
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = re.sub(' *', '', title)

    preprocessed_grades.append(title.strip())
```

```
100%|██████████| 50000/50000 [00:00<00:00, 112463.08it/s]
```

In [21]:
```
project_data.drop("project_grade_category", axis=1, inplace=True)
project_data["filtered_grade_category"] = preprocessed_grades
```

1.5 Preparing data for models

In [22]:
```
project_data.columns
```

Out[22]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', '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',
       'filtered_grade_category'],
      dtype='object')
```

we are going to consider

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

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

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

Modifying DataSet (essay & project_title):

In [23]:
```
project_data['clean_essay'] = preprocessed_essays
project_data['clean_project_title'] = preprocessed_titles
project_data.drop(['essay'], axis=1, inplace=True)
project_data.drop(['project_title'], axis=1, inplace=True)
```

Calculate Sentiment Scores for the essays

In [24]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
analyser = SentimentIntensityAnalyzer()

#http://t-redactyl.io/blog/2017/04/using-vader-to-handle-sentiment-analysis-with-social-media-text.html
neg = []
pos = []
neu = []
compound = []

for a in tqdm(project_data["clean_essay"]) :
    b = analyser.polarity_scores(a)['neg']
    c = analyser.polarity_scores(a)['pos']
    d = analyser.polarity_scores(a)['neu']
    e = analyser.polarity_scores(a)['compound']
    neg.append(b)
    pos.append(c)
    neu.append(d)
    compound.append(e)
```

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/amit/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
100%|██████████| 50000/50000 [09:12<00:00, 90.53it/s]
```

In [25]:
```
project_data["pos"] = pos
project_data["neg"] = neg
project_data["neu"] = neu
project_data["compound"] = compound
```

Essay word count

In [26]:
```
essay_count = []
for c in project_data["clean_essay"]:
    d = len(c.split())
    essay_count.append(d)

project_data["essay_count"] = essay_count
```

Project title word count

In [27]:
```
title_count = []
for a in project_data["clean_project_title"]:
    b = len(a.split())
    title_count.append(b)

project_data["title_count"] = title_count
project_data.head(5)
```

Out[27]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_essay_1	project_essay_2	project_ess
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	2016-04-27 00:53:00	I recently read an article about giving studen...	I teach at a low-income (Title 1) school. Ever...	We n classroo that we ca as
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	My students crave challenge, they eat obstacle...	We are an urban, public k-5 elementary school....	With the commor standard hav
29891	146723	p099708	c0a28c79fe8ad5810da49de47b3fb491	Mrs.	CA	2016-04-27 01:10:09	It's the end of the school year. Routines have...	My students desire challenges, movement, and c...	I will d different using specif
23374	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	Ms.	CA	2016-04-27 02:04:15	Never has society so rapidly changed. Technolo...	Our Language Arts and Social Justice Magnet Sc...	\"Is it my Ms. K? Whe I going to
49228	57854	p099430	4000cfe0c8b2df75a218347c1765e283	Ms.	IL	2016-04-27 07:19:44	My students yearn for a classroom environment ...	I have the privilege of teaching an incredible...	Ideally, I v love to right into

5 rows × 24 columns

Splitting

In [28]:
```
from sklearn.model_selection import train_test_split
y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
X = project_data


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

1.5.1 Vectorizing Categorical data

clean_categories:

In [29]:
```
vectorizer = CountVectorizer()
vectorizer.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_cc_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_cv_cc_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_cc_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_cc_ohe.shape, y_train.shape)
print(X_cv_cc_ohe.shape, y_cv.shape)
print(X_test_cc_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language', 'math_science',
'music_arts', 'specialneeds', 'warmth']
```

clean_subcategories

In [30]:
```
vectorizer = CountVectorizer()
vectorizer.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_csc_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_cv_csc_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_csc_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_csc_ohe.shape, y_train.shape)
print(X_cv_csc_ohe.shape, y_cv.shape)
print(X_test_csc_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
```

```
After vectorizations
(22445, 30) (22445,)
(11055, 30) (11055,)
(16500, 30) (16500,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government', 'college_careerprep', 'communi
tyservice', 'earlydevelopment', 'economics', 'environmentalscience', 'esl', 'extracurricular', 'financiallit
eracy', 'foreignlanguages', 'gym_fitness', 'health_lifescience', 'health_wellness', 'history_geography', 'li
teracy', 'literature_writing', 'mathematics', 'music', 'nutritioneducation', 'other', 'parentinvolvement',
'performingarts', 'socialsciences', 'specialneeds', 'teamsports', 'visualarts', 'warmth']
```

school_state

In [31]:
```
vectorizer = CountVectorizer()
vectorizer.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.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer.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.get_feature_names())
```

```
After vectorizations
(22445, 51) (22445,)
(11055, 51) (11055,)
(16500, 51) (16500,)
['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', 'ny', 'oh',
'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
```

teacher_prefix

In [32]:
```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values.astype('U')) # fit has to happen only on train data

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

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.get_feature_names())
```

```
After vectorizations
(22445, 6) (22445,)
(11055, 6) (11055,)
(16500, 6) (16500,)
['dr', 'mr', 'mrs', 'ms', 'nan', 'teacher']
```

project_grade_category

In [33]:
```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['filtered_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.transform(X_train['filtered_grade_category'].values)
X_cv_grade_ohe = vectorizer.transform(X_cv['filtered_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['filtered_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.get_feature_names())
```

```
After vectorizations
(22445, 4) (22445,)
(11055, 4) (11055,)
(16500, 4) (16500,)
['grades35', 'grades68', 'grades912', 'gradesprek2']
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

essays

In [34]:
```
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,2),max_features=5000)
vectorizer.fit(X_train['clean_essay'].values) # fit has to happen only on train data

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

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

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

project_title

In [35]:
```
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,2),max_features=5000)
vectorizer.fit(X_train['clean_project_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_pt_bow = vectorizer.transform(X_train['clean_project_title'].values)
X_cv_pt_bow = vectorizer.transform(X_cv['clean_project_title'].values)
X_test_pt_bow = vectorizer.transform(X_test['clean_project_title'].values)

print("After vectorizations")
print(X_train_pt_bow.shape, y_train.shape)
print(X_cv_pt_bow.shape, y_cv.shape)
print(X_test_pt_bow.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1850) (22445,)
(11055, 1850) (11055,)
(16500, 1850) (16500,)
====================================================================================================
```

1.5.2.2 TFIDF vectorizer

essays

In [36]:
```
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,2),max_features=5000)
X_train_essay_tfidf = vectorizer.fit_transform(X_train['clean_essay'].values)
X_cv_essay_tfidf = vectorizer.transform(X_cv['clean_essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['clean_essay'].values)



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

```
Shape of matrix after one hot encodig  (22445, 5000)
Shape of matrix after one hot encodig  (11055, 5000)
Shape of matrix after one hot encodig  (16500, 5000)
```

project_title

In [37]:
```
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,2),max_features=5000)
X_train_pt_tfidf = vectorizer.fit_transform(X_train['clean_project_title'].values)# fit has to happen only or
X_cv_pt_tfidf = vectorizer.transform(X_cv['clean_project_title'].values)
X_test_pt_tfidf = vectorizer.transform(X_test['clean_project_title'].values)



print("Shape of matrix after one hot encodig ",X_train_pt_tfidf.shape)
print("Shape of matrix after one hot encodig ",X_cv_pt_tfidf.shape)
print("Shape of matrix after one hot encodig ",X_test_pt_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (22445, 1850)
Shape of matrix after one hot encodig  (11055, 1850)
Shape of matrix after one hot encodig  (16500, 1850)
```

1.5.2.3 Using Pretrained Models: Avg W2V

1.5.2.3.1 essays

train

In [38]:
```
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [39]:
```
train_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    train_avg_w2v_vectors.append(vector)

print(len(train_avg_w2v_vectors))
print(len(train_avg_w2v_vectors[0]))
#print(w2v_model.wv.most_similar('worst'))
```

```
100%|██████████| 22445/22445 [00:11<00:00, 1889.06it/s]

22445
300
```

cross validate

In [40]:
```
cv_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['clean_essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    cv_avg_w2v_vectors.append(vector)

print(len(cv_avg_w2v_vectors))
print(len(cv_avg_w2v_vectors[0]))
```

```
100%|██████████| 11055/11055 [00:05<00:00, 2008.79it/s]

11055
300
```

test

In [41]:
```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    test_avg_w2v_vectors.append(vector)

print(len(test_avg_w2v_vectors))
print(len(test_avg_w2v_vectors[0]))
```

```
100%|██████████| 16500/16500 [00:07<00:00, 2121.23it/s]

16500
300
```

1.5.2.3.2 project_title

train

In [42]:
```
# average Word2Vec
# compute average word2vec for each review.
train_title_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    train_title_avg_w2v_vectors.append(vector)

print(len(train_title_avg_w2v_vectors))
print(len(train_title_avg_w2v_vectors[0]))
```

```
100%|██████████| 22445/22445 [00:00<00:00, 38070.13it/s]

22445
300
```

cross validate

In [43]:
```
# average Word2Vec
# compute average word2vec for each review.
cv_title_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['clean_project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    cv_title_avg_w2v_vectors.append(vector)

print(len(cv_title_avg_w2v_vectors))
print(len(cv_title_avg_w2v_vectors[0]))
```

```
100%|██████████| 11055/11055 [00:00<00:00, 41819.38it/s]

11055
300
```

test

In [44]:
```
# average Word2Vec
# compute average word2vec for each review.
test_title_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    test_title_avg_w2v_vectors.append(vector)

print(len(test_title_avg_w2v_vectors))
print(len(test_title_avg_w2v_vectors[0]))
```

```
100%|██████████| 16500/16500 [00:00<00:00, 40849.87it/s]

16500
300
```

1.5.2.4 Using Pretrained Models: TFIDF weighted W2V

1.5.2.4.1 essays

train

In [45]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['clean_essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
print(tfidf_model.idf_)
```

```
[ 7.30529564  5.8598124   9.92025541 ... 10.32572052 10.32572052
 10.32572052]
```

In [46]:
```
# average Word2Vec
# compute average word2vec for each review.
train_essay_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_essay_tfidf_w2v_vectors.append(vector)

print(len(train_essay_tfidf_w2v_vectors))
print(len(train_essay_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 22445/22445 [01:05<00:00, 344.42it/s]

22445
300
```

cross valildate

In [47]:
```
# average Word2Vec
# compute average word2vec for each review.
cv_essay_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['clean_essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    cv_essay_tfidf_w2v_vectors.append(vector)

print(len(cv_essay_tfidf_w2v_vectors))
print(len(cv_essay_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 11055/11055 [00:31<00:00, 355.62it/s]

11055
300
```

test

In [48]:
```
# average Word2Vec
# compute average word2vec for each review.
test_essay_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_essay_tfidf_w2v_vectors.append(vector)

print(len(test_essay_tfidf_w2v_vectors))
print(len(test_essay_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 16500/16500 [00:47<00:00, 344.48it/s]

16500
300
```

1.5.2.4.2 project_title

train

In [49]:
```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['clean_project_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [50]:
```
# average Word2Vec
# compute average word2vec for each review.
train_title_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_title_tfidf_w2v_vectors.append(vector)

print(len(train_title_tfidf_w2v_vectors))
print(len(train_title_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 22445/22445 [00:01<00:00, 18006.15it/s]

22445
300
```

cross validate

In [51]:
```
# average Word2Vec
# compute average word2vec for each review.
cv_title_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['clean_project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    cv_title_tfidf_w2v_vectors.append(vector)

print(len(cv_title_tfidf_w2v_vectors))
print(len(cv_title_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 11055/11055 [00:00<00:00, 17922.38it/s]

11055
300
```

test

In [52]:
```
# average Word2Vec
# compute average word2vec for each review.
test_title_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_title_tfidf_w2v_vectors.append(vector)

print(len(test_title_tfidf_w2v_vectors))
print(len(test_title_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 16500/16500 [00:00<00:00, 18136.34it/s]

16500
300
```

1.5.3 Vectorizing Numerical features

In [53]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')
```

1.5.3.1 Price

In [54]:
```
from sklearn.preprocessing import Normalizer
price_scalar = Normalizer()

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

X_train_price_std= price_scalar.transform(X_train['price'].values.reshape(1,-1))
X_cv_price_std = price_scalar.transform(X_cv['price'].values.reshape(1,-1))
X_test_price_std = price_scalar.transform(X_test['price'].values.reshape(1,-1))

X_train_price_std = np.transpose(X_train_price_std)
X_cv_price_std = np.transpose(X_cv_price_std)
X_test_price_std = np.transpose(X_test_price_std)


print("After standardisation")
print(X_train_price_std.shape, y_train.shape)
print(X_cv_price_std.shape, y_cv.shape)
print(X_test_price_std.shape, y_test.shape)
print("="*100)
```

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

1.5.3.2 Quantity

In [55]:
```
quantity_norm = Normalizer()

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

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

X_train_quantity_norm = np.transpose(X_train_quantity_norm)
X_cv_quantity_norm = np.transpose(X_cv_quantity_norm)
X_test_quantity_norm = np.transpose(X_test_quantity_norm)

print("After standardisation")
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 standardisation
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

1.5.3.3 Teacher number of previously posted projects

In [56]:
```
warnings.filterwarnings("ignore")
tnpp_scalar = Normalizer()

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

X_train_tnppp = tnpp_scalar.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(
X_cv_tnppp = tnpp_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))
X_test_tnppp = tnpp_scalar.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(1,

X_train_tnppp = np.transpose(X_train_tnppp)
X_cv_tnppp = np.transpose(X_cv_tnppp)
X_test_tnppp = np.transpose(X_test_tnppp)



print("After vectorizations")
print(X_train_tnppp.shape, y_train.shape)
print(X_cv_tnppp.shape, y_cv.shape)
print(X_test_tnppp.shape, y_test.shape)
print("="*100)
```

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

1.5.3.4 Sentiment scores of essays

In [57]:
```
pos_norm = Normalizer()

pos_norm.fit(X_train['pos'].values.reshape(1,-1))

X_train_pos_norm= pos_norm.transform(X_train['pos'].values.reshape(1,-1))
X_cv_pos_norm = pos_norm.transform(X_cv['pos'].values.reshape(1,-1))
X_test_pos_norm = pos_norm.transform(X_test['pos'].values.reshape(1,-1))

X_train_pos_norm = np.transpose(X_train_pos_norm)
X_cv_pos_norm = np.transpose(X_cv_pos_norm)
X_test_pos_norm = np.transpose(X_test_pos_norm)

print("After normalisation")
print(X_train_pos_norm.shape, y_train.shape)
print(X_cv_pos_norm.shape, y_cv.shape)
print(X_test_pos_norm.shape, y_test.shape)
print("="*100)
```

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

In [58]:
```
neg_norm = Normalizer()

neg_norm.fit(X_train['neg'].values.reshape(1,-1))

X_train_neg_norm= pos_norm.transform(X_train['neg'].values.reshape(1,-1))
X_cv_neg_norm = pos_norm.transform(X_cv['neg'].values.reshape(1,-1))
X_test_neg_norm = pos_norm.transform(X_test['neg'].values.reshape(1,-1))

X_train_neg_norm = np.transpose(X_train_neg_norm)
X_cv_neg_norm = np.transpose(X_cv_neg_norm)
X_test_neg_norm = np.transpose(X_test_neg_norm)

print("After normalisation")
print(X_train_neg_norm.shape, y_train.shape)
print(X_cv_neg_norm.shape, y_cv.shape)
print(X_test_neg_norm.shape, y_test.shape)
print("="*100)
```

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

In [59]:
```
neutral_norm = Normalizer()

neutral_norm.fit(X_train['neu'].values.reshape(1,-1))

X_train_neu_norm= neutral_norm.transform(X_train['neu'].values.reshape(1,-1))
X_cv_neu_norm = neutral_norm.transform(X_cv['neu'].values.reshape(1,-1))
X_test_neu_norm = neutral_norm.transform(X_test['neu'].values.reshape(1,-1))

X_train_neu_norm = np.transpose(X_train_neu_norm)
X_cv_neu_norm = np.transpose(X_cv_neu_norm)
X_test_neu_norm = np.transpose(X_test_neu_norm)

print("After normalisation")
print(X_train_neu_norm.shape, y_train.shape)
print(X_cv_neu_norm.shape, y_cv.shape)
print(X_test_neu_norm.shape, y_test.shape)
print("="*100)
```

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

In [60]:
```
compound_norm = Normalizer()

compound_norm.fit(X_train['compound'].values.reshape(1,-1))

X_train_compound_norm= compound_norm.transform(X_train['compound'].values.reshape(1,-1))
X_cv_compound_norm = compound_norm.transform(X_cv['compound'].values.reshape(1,-1))
X_test_compound_norm = compound_norm.transform(X_test['compound'].values.reshape(1,-1))

X_train_compound_norm = np.transpose(X_train_compound_norm)
X_cv_compound_norm = np.transpose(X_cv_compound_norm)
X_test_compound_norm = np.transpose(X_test_compound_norm)

print("After normalisation")
print(X_train_compound_norm.shape, y_train.shape)
print(X_cv_compound_norm.shape, y_cv.shape)
print(X_test_compound_norm.shape, y_test.shape)
print("="*100)
```

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

1.5.3.5 Essay word count

In [61]:
```
essay_count_norm = Normalizer()

essay_count_norm.fit(X_train['essay_count'].values.reshape(1,-1))

X_train_essay_count_norm = essay_count_norm.transform(X_train['essay_count'].values.reshape(1,-1))
X_cv_essay_count_norm = essay_count_norm.transform(X_cv['essay_count'].values.reshape(1,-1))
X_test_essay_count_norm = essay_count_norm.transform(X_test['essay_count'].values.reshape(1,-1))

X_train_essay_count_norm = np.transpose(X_train_essay_count_norm)
X_cv_essay_count_norm = np.transpose(X_cv_essay_count_norm)
X_test_essay_count_norm = np.transpose(X_test_essay_count_norm)

print("After normalisation")
print(X_train_essay_count_norm.shape, y_train.shape)
print(X_cv_essay_count_norm.shape, y_cv.shape)
print(X_test_essay_count_norm.shape, y_test.shape)
print("="*100)
```

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

1.5.3.6 Project title word count

In [62]:
```
title_count_norm = Normalizer()

title_count_norm.fit(X_train['title_count'].values.reshape(1,-1))

X_train_title_count_norm= title_count_norm.transform(X_train['title_count'].values.reshape(1,-1))
X_cv_title_count_norm = title_count_norm.transform(X_cv['title_count'].values.reshape(1,-1))
X_test_title_count_norm = title_count_norm.transform(X_test['title_count'].values.reshape(1,-1))

X_train_title_count_norm = np.transpose(X_train_title_count_norm)
X_cv_title_count_norm = np.transpose(X_cv_title_count_norm)
X_test_title_count_norm = np.transpose(X_test_title_count_norm)

print("After normalisation")
print(X_train_title_count_norm.shape, y_train.shape)
print(X_cv_title_count_norm.shape, y_cv.shape)
print(X_test_title_count_norm.shape, y_test.shape)
print("="*100)
```

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

1.5.4 Merging features:

SET 1

In [63]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr1 = hstack((X_train_cc_ohe, X_train_csc_ohe, X_train_grade_ohe, X_train_price_std, X_train_tnppp, X_trair
X_cv1 = hstack((X_cv_cc_ohe, X_cv_csc_ohe, X_cv_grade_ohe, X_cv_price_std, X_cv_tnppp, X_cv_essay_bow, X_cv_p
X_te1 = hstack((X_test_cc_ohe, X_test_csc_ohe, X_test_grade_ohe, X_test_price_std, X_test_tnppp, X_test_essay

print("Final Data matrix")
print(X_tr1.shape, y_train.shape)
print(X_cv1.shape, y_cv.shape)
print(X_te1.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 6895) (22445,)
(11055, 6895) (11055,)
(16500, 6895) (16500,)
====================================================================================================
```

SET2

In [64]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

X_tr2 = hstack((X_train_cc_ohe, X_train_csc_ohe, X_train_grade_ohe, X_train_price_std, X_train_tnppp, X_trair
X_cv2 = hstack((X_cv_cc_ohe, X_cv_csc_ohe, X_cv_grade_ohe, X_cv_price_std, X_cv_tnppp, X_cv_essay_tfidf, X_cv
X_te2 = hstack((X_test_cc_ohe, X_test_csc_ohe, X_test_grade_ohe, X_test_price_std, X_test_tnppp, X_test_essay

print("Final Data matrix")
print(X_tr2.shape, y_train.shape)
print(X_cv2.shape, y_cv.shape)
print(X_te2.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 6895) (22445,)
(11055, 6895) (11055,)
(16500, 6895) (16500,)
====================================================================================================
```

SET3

In [65]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

X_tr3 = hstack((X_train_cc_ohe, X_train_csc_ohe, X_train_grade_ohe, X_train_price_std, X_train_tnppp, train_a
X_cv3 = hstack((X_cv_cc_ohe, X_cv_csc_ohe, X_cv_grade_ohe, X_cv_price_std, X_cv_tnppp, cv_avg_w2v_vectors, cv
X_te3 = hstack((X_test_cc_ohe, X_test_csc_ohe, X_test_grade_ohe, X_test_price_std, X_test_tnppp, test_avg_w2v

print("Final Data matrix")
print(X_tr3.shape, y_train.shape)
print(X_cv3.shape, y_cv.shape)
print(X_te3.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 645) (22445,)
(11055, 645) (11055,)
(16500, 645) (16500,)
====================================================================================================
```

SET4

In [66]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

X_tr4 = hstack((X_train_cc_ohe, X_train_csc_ohe, X_train_grade_ohe, X_train_price_std, X_train_tnppp, train_e
X_cv4 = hstack((X_cv_cc_ohe, X_cv_csc_ohe, X_cv_grade_ohe, X_cv_price_std, X_cv_tnppp, cv_essay_tfidf_w2v_vec
X_te4 = hstack((X_test_cc_ohe, X_test_csc_ohe, X_test_grade_ohe, X_test_price_std, X_test_tnppp, test_essay_t

print("Final Data matrix")
print(X_tr4.shape, y_train.shape)
print(X_cv4.shape, y_cv.shape)
print(X_te4.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 645) (22445,)
(11055, 645) (11055,)
(16500, 645) (16500,)
====================================================================================================
```

SET5

In [67]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

X_tr5 = hstack((X_train_state_ohe,X_train_cc_ohe, X_train_csc_ohe,X_train_grade_ohe,X_train_teacher_ohe,X_tra
X_cv5 = hstack((X_cv_state_ohe,X_cv_cc_ohe, X_cv_csc_ohe, X_cv_grade_ohe,X_cv_teacher_ohe,X_cv_quantity_norm,
X_te5 = hstack((X_test_state_ohe,X_test_cc_ohe, X_test_csc_ohe,X_test_grade_ohe,X_test_teacher_ohe,X_test_qua

print("Final Data matrix")
print(X_tr5.shape, y_train.shape)
print(X_cv5.shape, y_cv.shape)
print(X_te5.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(22445, 109) (22445,)
(11055, 109) (11055,)
(16500, 109) (16500,)
====================================================================================================
```

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/) 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/) 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)

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/)

Note: Data Leakage

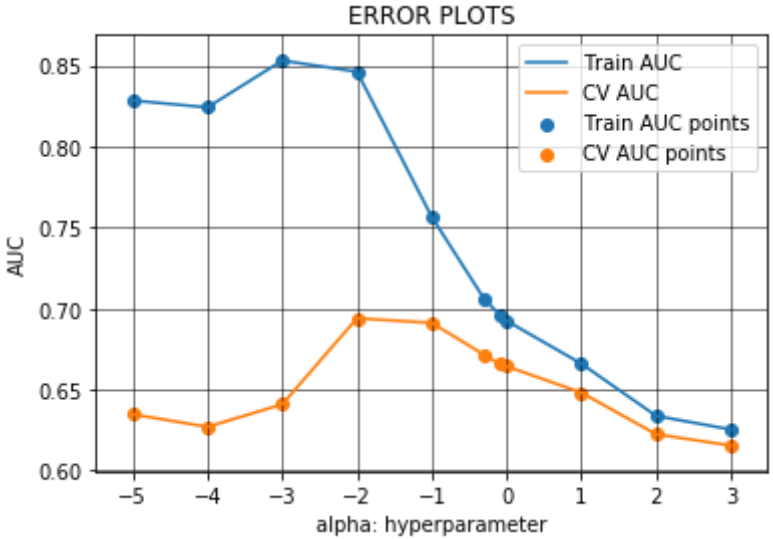
1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakag, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply 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)

2. Hyperparameter Tuning

2.1 Error plots using SET1 data

In [71]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier


SGD = SGDClassifier(loss= 'log', penalty= 'l2',class_weight = 'balanced')

parameters = {'alpha':[0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]}

clf = GridSearchCV(SGD, parameters, cv= 10, scoring='roc_auc',return_train_score=True)

clf.fit(X_tr1, y_train)

train_auc1= clf.cv_results_['mean_train_score']
train_auc_std1= clf.cv_results_['std_train_score']
cv_auc1 = clf.cv_results_['mean_test_score']
cv_auc_std1= clf.cv_results_['std_test_score']
```

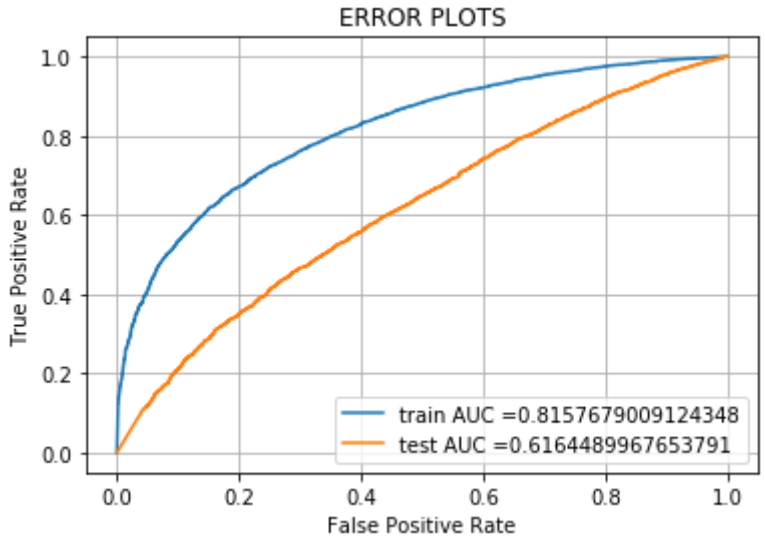
In [72]:
```
alphas = [0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]

plt.plot(np.log10(alphas), train_auc1, label='Train AUC')
plt.plot(np.log10(alphas), cv_auc1, label='CV AUC')

plt.scatter(np.log10(alphas), train_auc1, label='Train AUC points')
plt.scatter(np.log10(alphas), cv_auc1, label='CV AUC points')


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



In [73]:
```
best_alpha1=clf.best_params_
print(best_alpha1)
```

```
{'alpha': 0.01}
```

In [74]:
```
best_alfa = best_alpha1['alpha']
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

model1 = SGDClassifier(loss= 'log', penalty= 'l2', class_weight = 'balanced')
model1.fit(X_tr1,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_predicted_tr1 = model1.predict_proba(X_tr1)[:,1]
y_predicted_te1 = model1.predict_proba(X_te1)[:,1]

train_fpr1, train_tpr1, tr_thresholds1 = roc_curve(y_train, y_predicted_tr1)
test_fpr1, test_tpr1, te_thresholds1 = roc_curve(y_test, y_predicted_te1)

plt.plot(train_fpr1, train_tpr1, label="train AUC ="+str(auc(train_fpr1, train_tpr1)))
plt.plot(test_fpr1, test_tpr1, label="test AUC ="+str(auc(test_fpr1, test_tpr1)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

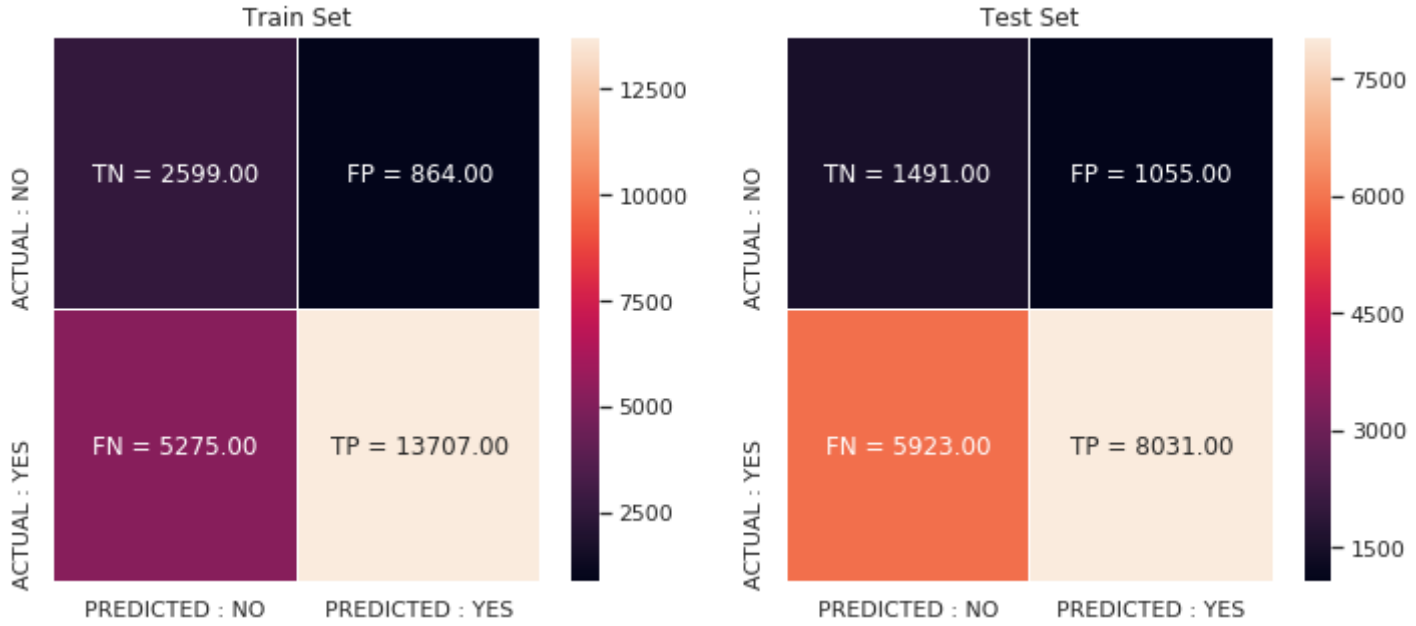


Confusion Matrix

In [75]:
```
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 [76]:
```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
print('Train Set:')
con_m_train = confusion_matrix(y_train, predict(y_predicted_tr1, tr_thresholds1, train_fpr1, train_tpr1))
print('Test Set:')
con_m_test = confusion_matrix(y_test, predict(y_predicted_te1, te_thresholds1, test_fpr1, test_tpr1))

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

fig, ax = plt.subplots(1,2, figsize=(12,5))

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


sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTU

sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTUA

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

plt.show()
```

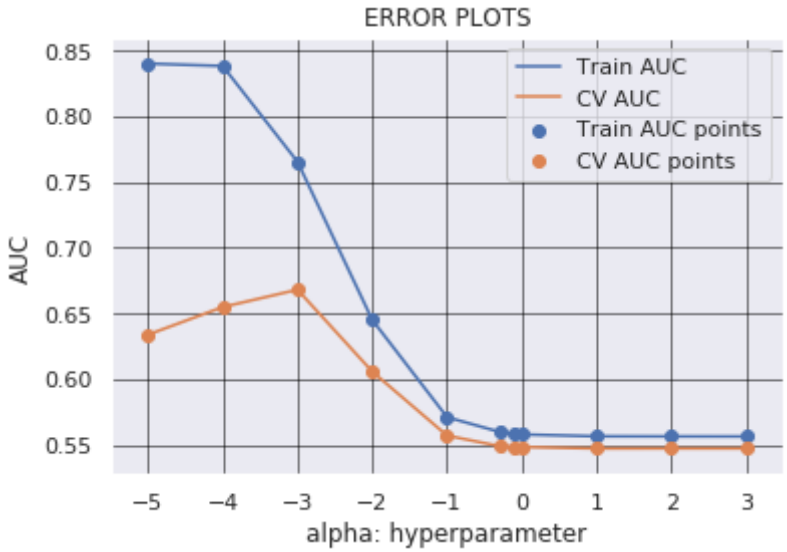
```
Train Set:
the maximum value of tpr*(1-fpr) 0.5419437743853449 for threshold 0.0
Test Set:
the maximum value of tpr*(1-fpr) 0.3370467559158861 for threshold 0.0
```



2.2 Error plots using SET2 data

In [77]:
```
SGD = SGDClassifier(loss= 'log', penalty= 'l2',class_weight = 'balanced')
parameters = {'alpha':[0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]}

clf = GridSearchCV(SGD, parameters, cv= 10, scoring='roc_auc',return_train_score=True)

clf.fit(X_tr2, y_train)

train_auc2= clf.cv_results_['mean_train_score']
train_auc_std2= clf.cv_results_['std_train_score']
cv_auc2 = clf.cv_results_['mean_test_score']
cv_auc_std2= clf.cv_results_['std_test_score']
```

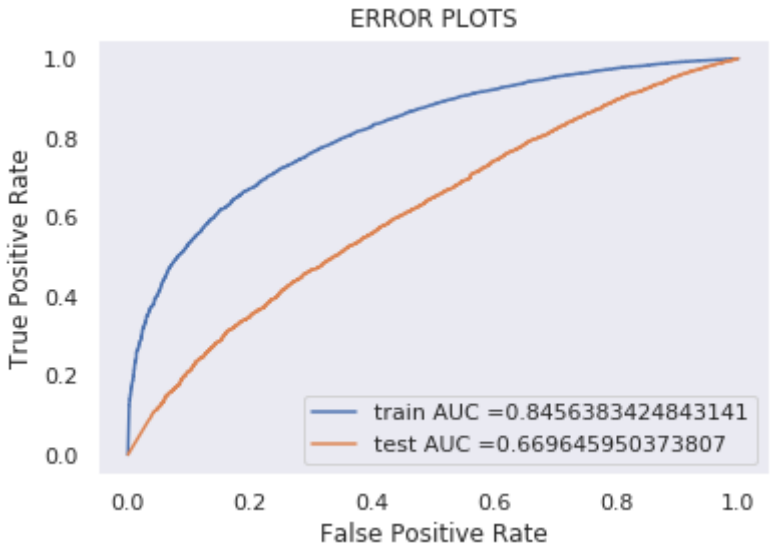
In [78]:
```
alphas = [0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]

plt.plot(np.log10(alphas), train_auc2, label='Train AUC')
plt.plot(np.log10(alphas), cv_auc2, label='CV AUC')

plt.scatter(np.log10(alphas), train_auc2, label='Train AUC points')
plt.scatter(np.log10(alphas), cv_auc2, label='CV AUC points')


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



In [79]:
```
best_alpha2=clf.best_params_
print(best_alpha2)
```

```
{'alpha': 0.001}
```

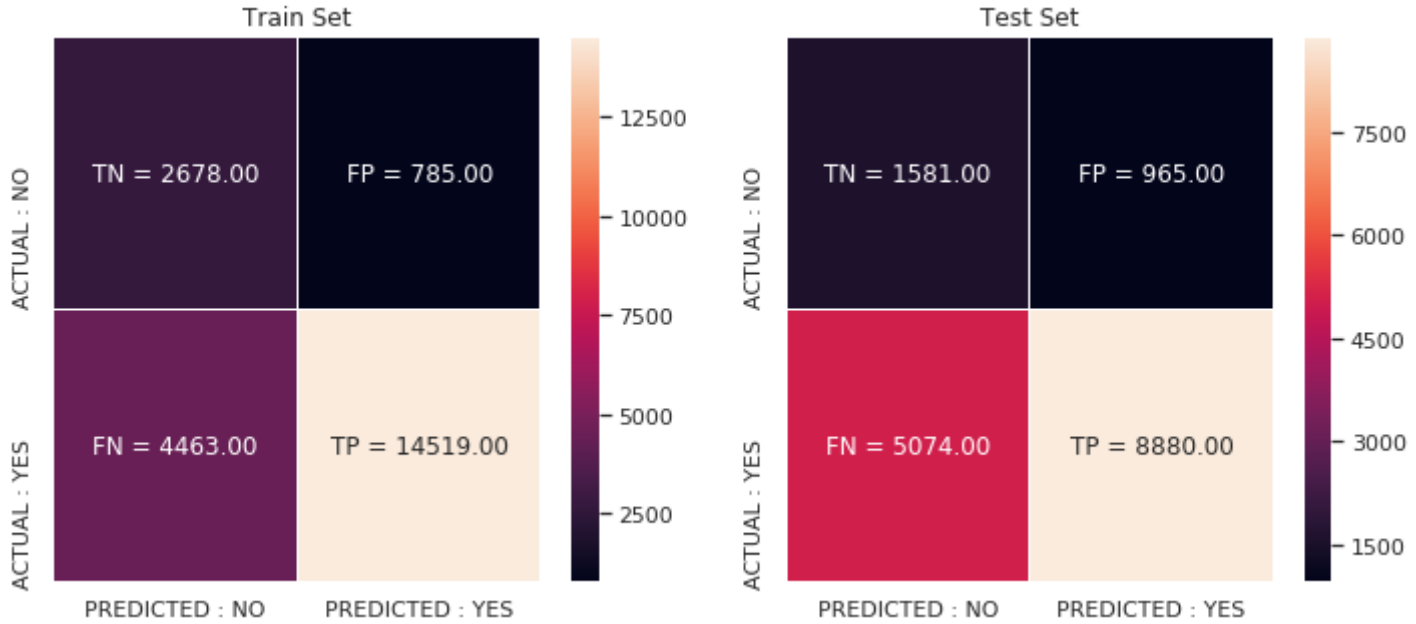
In [80]:
```
best_alfa = best_alpha2['alpha']
model2 = SGDClassifier(loss= 'log', penalty= 'l2',class_weight = 'balanced')
model2.fit(X_tr2,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs


y_predicted_tr2 = model2.predict_proba(X_tr2)[:,1]
y_predicted_te2 = model2.predict_proba(X_te2)[:,1]

train_fpr2, train_tpr2, tr_thresholds2 = roc_curve(y_train, y_predicted_tr2)
test_fpr2, test_tpr2, te_thresholds2 = roc_curve(y_test, y_predicted_te2)

plt.plot(train_fpr1, train_tpr1, label="train AUC ="+str(auc(train_fpr2, train_tpr2)))
plt.plot(test_fpr1, test_tpr1, label="test AUC ="+str(auc(test_fpr2, test_tpr2)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



Confusion Matrix

In [81]:
```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
print('Train Set:')
con_m_train = confusion_matrix(y_train, predict(y_predicted_tr2, tr_thresholds2, train_fpr2, train_tpr2))
print('Test Set:')
con_m_test = confusion_matrix(y_test, predict(y_predicted_te2, te_thresholds2, test_fpr2, test_tpr2))

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

fig, ax = plt.subplots(1,2, figsize=(12,5))

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


sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTU

sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTUA

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

plt.show()
```

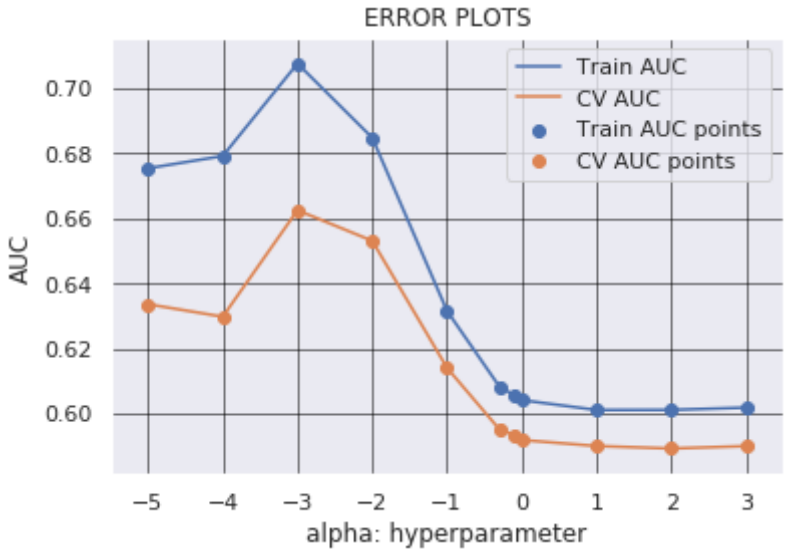
```
Train Set:
the maximum value of tpr*(1-fpr) 0.5914973691354879 for threshold 0.509
Test Set:
the maximum value of tpr*(1-fpr) 0.3951734129004953 for threshold 0.566
```



2.3 Error plots using SET3 data

In [82]:
```
SGD = SGDClassifier(loss= 'log', penalty= 'l2', class_weight = 'balanced')

parameters = {'alpha':[0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]}

clf = GridSearchCV(SGD, parameters, cv= 10, scoring='roc_auc',return_train_score=True)

clf.fit(X_tr3, y_train)

train_auc3= clf.cv_results_['mean_train_score']
train_auc_std3= clf.cv_results_['std_train_score']
cv_auc3 = clf.cv_results_['mean_test_score']
cv_auc_std3= clf.cv_results_['std_test_score']
```

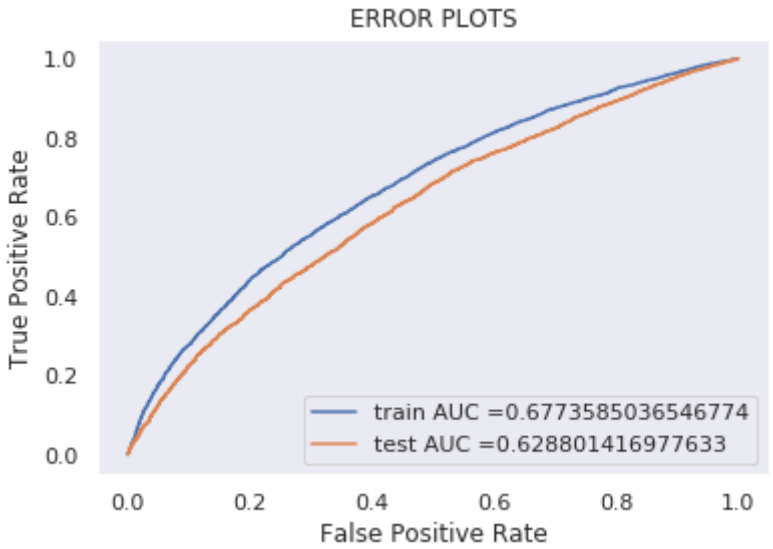
In [83]:
```
alphas = [0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]

plt.plot(np.log10(alphas), train_auc3, label='Train AUC')
plt.plot(np.log10(alphas), cv_auc3, label='CV AUC')

plt.scatter(np.log10(alphas), train_auc3, label='Train AUC points')
plt.scatter(np.log10(alphas), cv_auc3, label='CV AUC points')


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



In [84]:
```
best_alpha3 = clf.best_params_
print(best_alpha3)
```

```
{'alpha': 0.001}
```

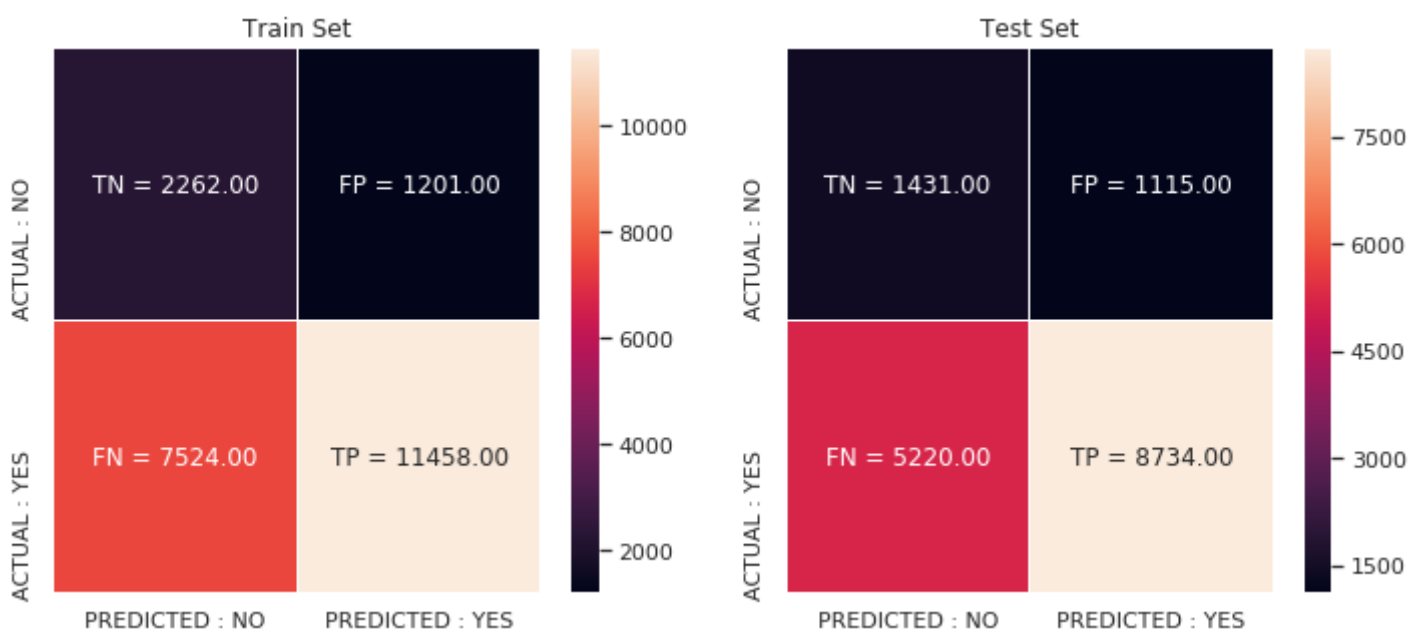
In [85]:
```
best_alfa = best_alpha3['alpha']
model3 = SGDClassifier(loss= 'log', penalty= 'l2', class_weight = 'balanced')
model3.fit(X_tr3,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs


y_predicted_tr3 = model3.predict_proba(X_tr3)[:,1]
y_predicted_te3 = model3.predict_proba(X_te3)[:,1]

train_fpr3, train_tpr3, tr_thresholds3 = roc_curve(y_train, y_predicted_tr3)
test_fpr3, test_tpr3, te_thresholds3 = roc_curve(y_test, y_predicted_te3)

plt.plot(train_fpr3, train_tpr3, label="train AUC ="+str(auc(train_fpr3, train_tpr3)))
plt.plot(test_fpr3, test_tpr3, label="test AUC ="+str(auc(test_fpr3, test_tpr3)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



Confusion Matrix

In [86]:
```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
print('Train Set:')
con_m_train = confusion_matrix(y_train, predict(y_predicted_tr3, tr_thresholds3, train_fpr3, train_tpr3))
print('Test Set:')
con_m_test = confusion_matrix(y_test, predict(y_predicted_te3, te_thresholds3, test_fpr3, test_tpr3))

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

fig, ax = plt.subplots(1,2, figsize=(12,5))

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


sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTU

sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTUA

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

plt.show()
```

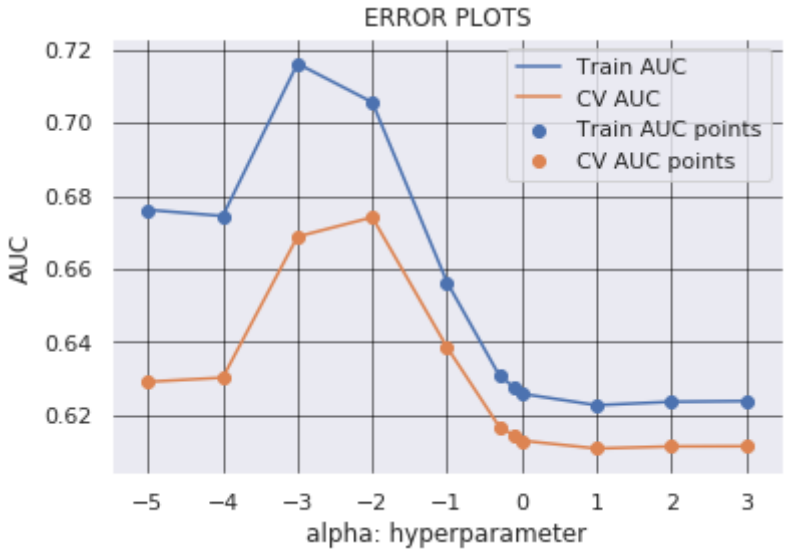
```
Train Set:
the maximum value of tpr*(1-fpr) 0.3942820063921828 for threshold 0.635
Test Set:
the maximum value of tpr*(1-fpr) 0.3517998932864475 for threshold 0.569
```



2.4 Error plots using SET4 data

In [87]:
```
SGD = SGDClassifier(loss= 'log', penalty= 'l2', class_weight = 'balanced')

parameters = {'alpha':[0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]}

clf = GridSearchCV(SGD, parameters, cv= 10, scoring='roc_auc',return_train_score=True)

clf.fit(X_tr4, y_train)

train_auc4= clf.cv_results_['mean_train_score']
train_auc_std4= clf.cv_results_['std_train_score']
cv_auc4 = clf.cv_results_['mean_test_score']
cv_auc_std4= clf.cv_results_['std_test_score']
```

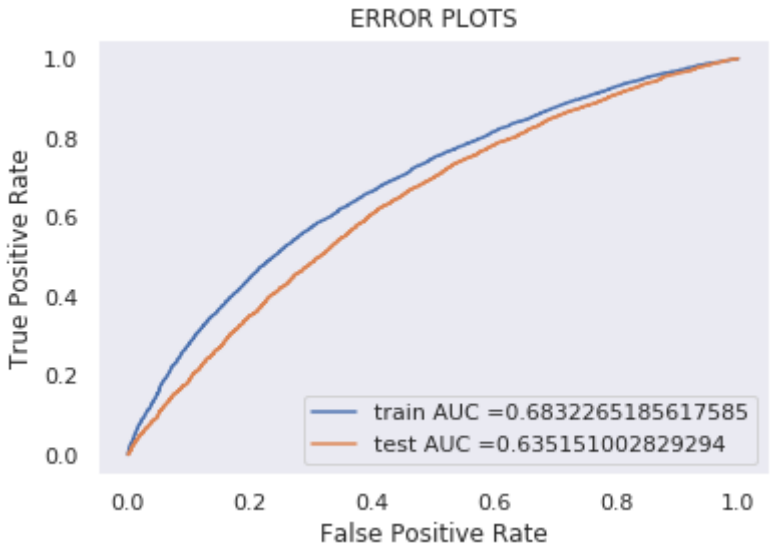
In [88]:
```
alphas = [0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]

plt.plot(np.log10(alphas), train_auc4, label='Train AUC')
plt.plot(np.log10(alphas), cv_auc4, label='CV AUC')

plt.scatter(np.log10(alphas), train_auc4, label='Train AUC points')
plt.scatter(np.log10(alphas), cv_auc4, label='CV AUC points')


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



In [89]:
```
best_alpha4 = clf.best_params_
print(best_alpha4)
```

```
{'alpha': 0.01}
```

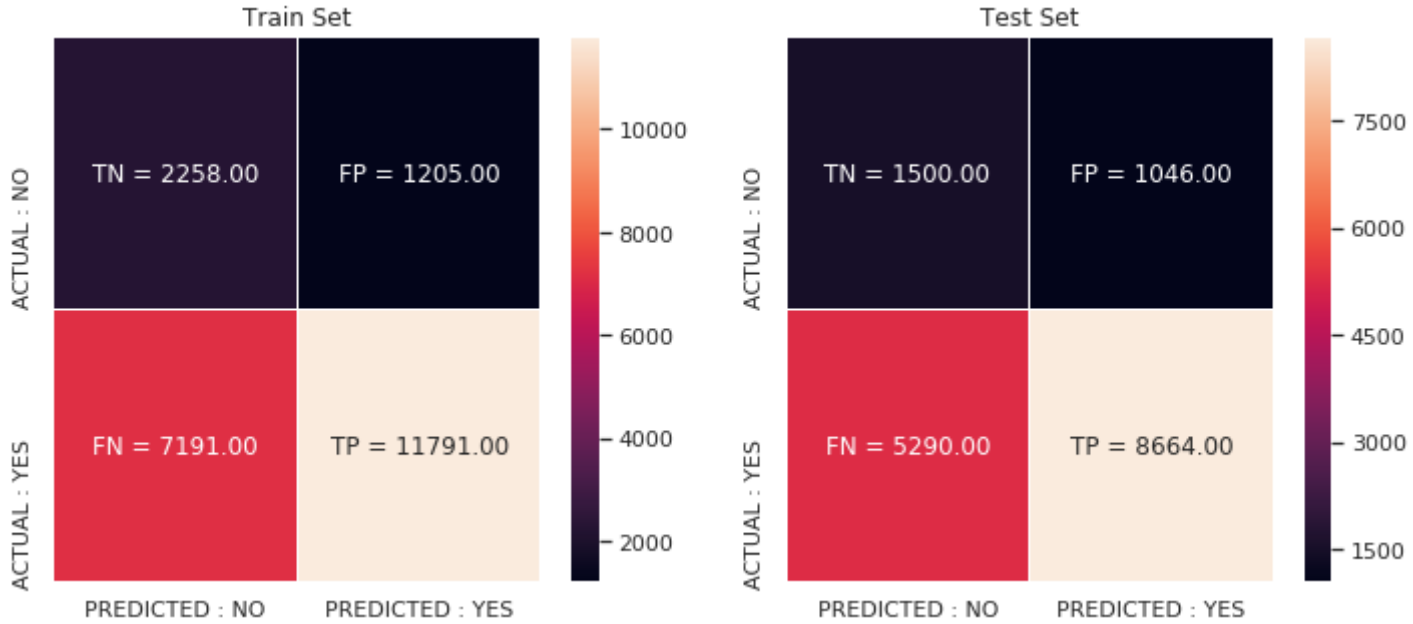
In [90]:
```
best_alfa = best_alpha4['alpha']
model4 = SGDClassifier(loss= 'log', penalty= 'l2', class_weight = 'balanced')
model4.fit(X_tr4,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs


y_predicted_tr4 = model4.predict_proba(X_tr4)[:,1]
y_predicted_te4 = model4.predict_proba(X_te4)[:,1]

train_fpr4, train_tpr4, tr_thresholds4 = roc_curve(y_train, y_predicted_tr4)
test_fpr4, test_tpr4, te_thresholds4 = roc_curve(y_test, y_predicted_te4)

plt.plot(train_fpr4, train_tpr4, label="train AUC ="+str(auc(train_fpr4, train_tpr4)))
plt.plot(test_fpr4, test_tpr4, label="test AUC ="+str(auc(test_fpr4, test_tpr4)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



Confusion Matrix

In [91]:
```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
print('Train Set:')
con_m_train = confusion_matrix(y_train, predict(y_predicted_tr4, tr_thresholds4, train_fpr4, train_tpr4))
print('Test Set:')
con_m_test = confusion_matrix(y_test, predict(y_predicted_te4, te_thresholds4, test_fpr4, test_tpr4))

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

fig, ax = plt.subplots(1,2, figsize=(12,5))

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


sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTU

sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTUA

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

plt.show()
```

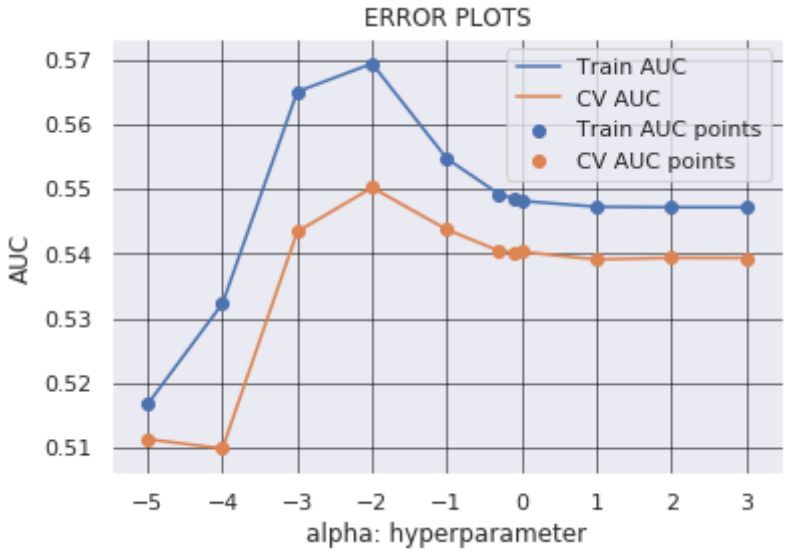
```
Train Set:
the maximum value of tpr*(1-fpr) 0.40502340119899605 for threshold 0.649
Test Set:
the maximum value of tpr*(1-fpr) 0.3658074825813601 for threshold 0.632
```



2.5 Error plots using SET5 data

In [92]:
```
SGD = SGDClassifier(loss= 'log', penalty= 'l2', class_weight = 'balanced')

parameters = {'alpha':[0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]}

clf = GridSearchCV(SGD, parameters, cv= 10, scoring='roc_auc',return_train_score=True)

clf.fit(X_tr5, y_train)

train_auc5= clf.cv_results_['mean_train_score']
train_auc_std5= clf.cv_results_['std_train_score']
cv_auc5 = clf.cv_results_['mean_test_score']
cv_auc_std5= clf.cv_results_['std_test_score']
```

In [93]:
```
alphas = [0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]

plt.plot(np.log10(alphas), train_auc5, label='Train AUC')
plt.plot(np.log10(alphas), cv_auc5, label='CV AUC')

plt.scatter(np.log10(alphas), train_auc5, label='Train AUC points')
plt.scatter(np.log10(alphas), cv_auc5, label='CV AUC points')


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



In [94]:
```
best_alpha5 = clf.best_params_
print(best_alpha5)
```

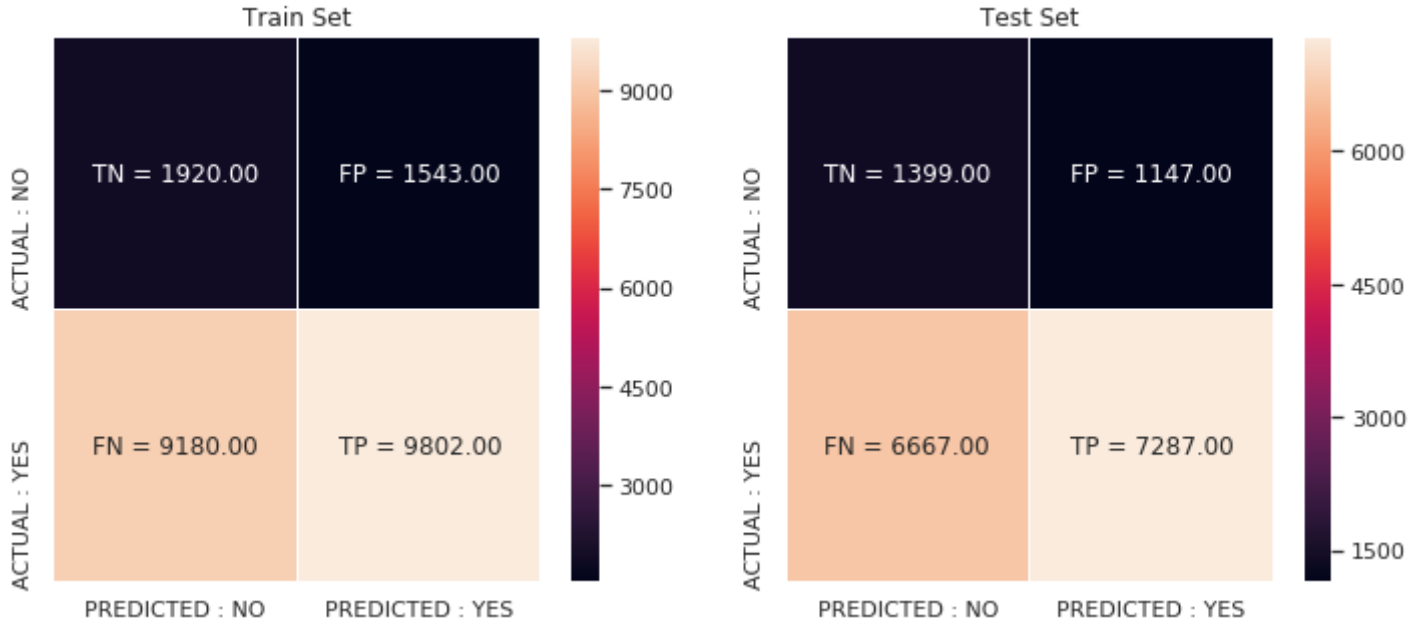
```
{'alpha': 0.01}
```

In [95]:
```
best_alfa = best_alpha5['alpha']
model5 = SGDClassifier(loss= 'log', penalty= 'l2', class_weight = 'balanced')
model5.fit(X_tr5,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_predicted_tr5 = model5.predict_proba(X_tr5)[:,1]
y_predicted_te5 = model5.predict_proba(X_te5)[:,1]

train_fpr5, train_tpr5, tr_thresholds5 = roc_curve(y_train, y_predicted_tr5)
test_fpr5, test_tpr5, te_thresholds5 = roc_curve(y_test, y_predicted_te5)

plt.plot(train_fpr5, train_tpr5, label="train AUC ="+str(auc(train_fpr5, train_tpr5)))
plt.plot(test_fpr5, test_tpr5, label="test AUC ="+str(auc(test_fpr5, test_tpr5)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

Confusion Matrix

In [96]:
```
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
print('Train Set:')
con_m_train = confusion_matrix(y_train, predict(y_predicted_tr5, tr_thresholds5, train_fpr5, train_tpr5))
print('Test Set:')
con_m_test = confusion_matrix(y_test, predict(y_predicted_te5, te_thresholds5, test_fpr5, test_tpr5))

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

fig, ax = plt.subplots(1,2, figsize=(12,5))

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


sns.heatmap(con_m_train, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTU

sns.heatmap(con_m_test, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'], yticklabels=['ACTUA

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

plt.show()
```

```
Train Set:
the maximum value of tpr*(1-fpr) 0.2863000779527807 for threshold 0.446
Test Set:
the maximum value of tpr*(1-fpr) 0.28695207268951595 for threshold 0.447
```


Train Set
TN = 1920.00
FP = 1543.00
FN = 9180.00
TP = 9802.00
ACTUAL : NO
ACTUAL : YES
PREDICTED : NO
PREDICTED : YES
9000
7500
6000
4500
3000
Test Set
TN = 1399.00
FP = 1147.00
FN = 6667.00
TP = 7287.00
ACTUAL : NO
ACTUAL : YES
PREDICTED : NO
PREDICTED : YES
6000
4500
3000
1500

In [98]:
```
# http://zetcode.com/python/prettytable/

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

x.add_row(["SET1_BOW", "Logistic Regression", 0.01, 0.64])
x.add_row(["SET2_TFIDF", "Logistic Regression", 0.0001, 0.66])
x.add_row(["SET3_AVGW2V", "Logistic Regression", 0.001, 0.62])
x.add_row(["SET4_TFIDFW2V", "Logistic Regression", 0.001, 0.63])
x.add_row(["SET5_WITHOUTTEXT", "Logistic Regression", 0.001, 0.54])
print(x)
```

```
+------------------+---------------------+-----------------------+----------+
|    Vectorizer    |        Model        | Alpha:Hyper Parameter | Test AUC |
+------------------+---------------------+-----------------------+----------+
|     SET1_BOW     | Logistic Regression |          0.01         |   0.64   |
|    SET2_TFIDF    | Logistic Regression |         0.0001        |   0.66   |
|   SET3_AVGW2V    | Logistic Regression |         0.001         |   0.62   |
|  SET4_TFIDFW2V   | Logistic Regression |         0.001         |   0.63   |
| SET5_WITHOUTTEXT | Logistic Regression |         0.001         |   0.54   |
+------------------+---------------------+-----------------------+----------+
```