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:** `p0`
`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. C enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for th 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_abbrevia **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories fo **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`

Feature	Description
project_resource_summary	An explanation of the resources needed for the project. **E** • My students need hands on literacy materia sensory needs!
project_essay_1	First application essay*
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Examp** 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed project bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix	Teacher's title. One of the following enumerated values: • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by th **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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:
```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'sc
hool_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]:
```
labels=project_data['project_is_approved']
project_data.drop(['project_is_approved'],axis=1,inplace=True)
```

In [5]:
```
labels=labels.head(50000)
```

In [6]:
```
project_data=project_data[0:50000]
```

In [7]:
```
project_data.head(1)
```

Out[7]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs	IN

Stratified Sampling: Splitting data into Train and Test

In [8]:
```
from sklearn.model_selection import train_test_split
project_data_train, project_data_test, labels_train, labels_test = train_test_
split(project_data, labels , test_size=0.33, stratify=labels)
print(project_data_train.shape)
print(project_data_test.shape)
print(labels_train.shape)
print(labels_test.shape)
```

```
(33500, 16)
(16500, 16)
(33500,)
(16500,)
```

In [9]:
```
project_data_train.head(2)
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
20583	152370	p048793	f2bbd023e6288ad1d47cca9db60b34b9	Mrs	NC
32043	138538	p007982	9d3c875fc42e2369cd0263b38b9421bb	Mrs	AR

In [10]:
```
labels=list(labels_train)
```

In [11]:
```
ids=list(project_data_train['id'])
```

In [12]:
```
data={'labels':labels, 'id':ids}

df=pd.DataFrame(data)

print(df.head(2))
```

```
   labels       id
0       1  p048793
1       0  p007982
```

In [13]:
```
project_data_train = pd.merge(project_data_train, df, on='id', how='left').res
et_index()

project_data_train.head(2)
```

Out[13]:

	index	Unnamed: 0	id	teacher_id	teacher_prefix	schoo
0	0	152370	p048793	f2bbd023e6288ad1d47cca9db60b34b9	Mrs	NC
1	1	138538	p007982	9d3c875fc42e2369cd0263b38b9421bb	Mrs	AR

In [14]:
```
project_data_train.drop(['Unnamed: 0','index'],axis=1,inplace=True)
```

In [15]:
```
project_data_train.head(2)
```

Out[15]:

	id	teacher_id	teacher_prefix	school_state	project_su
0	p048793	f2bbd023e6288ad1d47cca9db60b34b9	Mrs	NC	8/15/2016
1	p007982	9d3c875fc42e2369cd0263b38b9421bb	Mrs	AR	8/15/2016

preprocessing of `project_subject_categories` - Train Data

In [16]:
```
catogories = (project_data_train['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
 space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

In [17]:
```
unique_list = []
for x in cat_list:
    if x not in unique_list:
            unique_list.append(x)


#print(unique_list)

categories=pd.DataFrame({'clean_categories': unique_list})
categories=categories.sort_values(['clean_categories'], ascending=True).reset_
index()

print(categories.head(2))
```

```
   index              clean_categories
0      1                AppliedLearning
1     29  AppliedLearning Health_Sports
```

In [18]:
```
df1=project_data_train[['clean_categories','labels']][(project_data_train['lab
els']==1)]

print(df1.head(2))
```

```
             clean_categories  labels
0  Math_Science Health_Sports       1
3          Warmth Care_Hunger       1
```

In [19]:
```
df2=project_data_train[['clean_categories','labels']][(project_data_train['lab
els']==0)]
```

In [20]:
```
z =df1.groupby(['clean_categories'])['labels'].value_counts() /project_data_tr
ain.groupby(['clean_categories'])['labels'].count()

group_1=pd.DataFrame(z)

group_1=group_1.reset_index(drop=True)
print(group_1.head(2))
```

```
     labels
0  0.800173
1  0.835106
```

In [21]:
```
z1 =df2.groupby(['clean_categories'])['labels'].value_counts() /project_data_t
rain.groupby(['clean_categories'])['labels'].count()

group_0=pd.DataFrame(z1)

group_0=group_0.reset_index(drop=True)
print(group_0.head(2))
```

```
     labels
0  0.199827
1  0.164894
```

In [22]:
```
x1= df1.groupby(['clean_categories'])['labels'].value_counts()
class_1=pd.DataFrame(x1)
class_1=class_1.reset_index(drop=True)
print ( class_1.head(2))
```

```
   labels
0     925
1     157
```

In [23]:
```
x0= df2.groupby(['clean_categories'])['labels'].value_counts()
class_0=pd.DataFrame(x0)
class_0=class_0.reset_index(drop=True)
print ( class_0.head(2))
```

```
   labels
0     231
1      31
```

In [24]:
```
Response_Table = pd.concat([categories, class_0, class_1],axis=1)
```

In [25]:
```
#taken from https://stackoverflow.com/questions/24685012/pandas-dataframe-renaming-multiple-identically-named-columns

def df_column_uniquify(df):
    df_columns = df.columns
    new_columns = []
    for item in df_columns:
        counter = 0
        newitem = item
        while newitem in new_columns:
            counter += 1
            newitem = "{}_{}".format(item, counter)
        new_columns.append(newitem)
    df.columns = new_columns
    return df
```

In [26]:
```
Response_Table = df_column_uniquify(Response_Table)
```

In [27]:
```
Response_Table.rename(columns={'labels':'Class=0','labels_1':'Class=1'},inplace=True)

print("Response Table for Categories")

Response_Table
```

Response Table for Categories

Out[27]:

	index	clean_categories	Class=0	Class=1
0	1	AppliedLearning	231.0	925.0
1	29	AppliedLearning Health_Sports	31.0	157.0
2	30	AppliedLearning History_Civics	11.0	39.0
3	22	AppliedLearning Literacy_Language	102.0	586.0
4	19	AppliedLearning Math_Science	63.0	266.0
5	15	AppliedLearning Music_Arts	45.0	182.0
6	26	AppliedLearning SpecialNeeds	84.0	364.0
7	47	AppliedLearning Warmth Care_Hunger	2.0	3.0
8	7	Health_Sports	478.0	2672.0
9	39	Health_Sports AppliedLearning	12.0	60.0
10	41	Health_Sports History_Civics	1.0	15.0
11	27	Health_Sports Literacy_Language	50.0	217.0
12	17	Health_Sports Math_Science	18.0	62.0
13	11	Health_Sports Music_Arts	9.0	32.0
14	24	Health_Sports SpecialNeeds	53.0	361.0
15	40	Health_Sports Warmth Care_Hunger	1.0	6.0
16	12	History_Civics	103.0	464.0
17	45	History_Civics AppliedLearning	3.0	15.0
18	43	History_Civics Health_Sports	35.0	7.0
19	20	History_Civics Literacy_Language	16.0	410.0
20	28	History_Civics Math_Science	14.0	103.0
21	33	History_Civics Music_Arts	17.0	71.0
22	23	History_Civics SpecialNeeds	1002.0	52.0
23	6	Literacy_Language	30.0	6373.0
24	38	Literacy_Language AppliedLearning	4.0	158.0
25	34	Literacy_Language Health_Sports	25.0	21.0
26	18	Literacy_Language History_Civics	605.0	213.0
27	8	Literacy_Language Math_Science	97.0	3869.0
28	4	Literacy_Language Music_Arts	172.0	429.0
29	10	Literacy_Language SpecialNeeds	917.0	1016.0
30	32	Literacy_Language Warmth Care_Hunger	54.0	2.0
31	2	Math_Science	28.0	4229.0

	index	clean_categories	Class=0	Class=1
32	21	Math_Science AppliedLearning	27.0	308.0
33	0	Math_Science Health_Sports	95.0	100.0
34	9	Math_Science History_Civics	87.0	168.0
35	13	Math_Science Literacy_Language	94.0	594.0
36	14	Math_Science Music_Arts	243.0	419.0
37	25	Math_Science SpecialNeeds	1.0	484.0
38	46	Math_Science Warmth Care_Hunger	2.0	4.0
39	16	Music_Arts	4.0	1347.0
40	44	Music_Arts AppliedLearning	4.0	3.0
41	35	Music_Arts Health_Sports	1.0	7.0
42	42	Music_Arts History_Civics	250.0	3.0
43	36	Music_Arts SpecialNeeds	1.0	36.0
44	49	Music_Arts Warmth Care_Hunger	20.0	1016.0
45	5	SpecialNeeds	26.0	6.0
46	37	SpecialNeeds Health_Sports	NaN	73.0
47	31	SpecialNeeds Music_Arts	NaN	4.0
48	48	SpecialNeeds Warmth Care_Hunger	NaN	381.0
49	3	Warmth Care_Hunger	NaN	NaN

In [28]:
```
category_1 = pd.concat([categories,group_0,group_1],axis=1).reset_index()
category_1
```

Out[28]:

	level_0	index	clean_categories	labels	labels
0	0	1	AppliedLearning	0.199827	0.800173
1	1	29	AppliedLearning Health_Sports	0.164894	0.835106
2	2	30	AppliedLearning History_Civics	0.220000	0.780000
3	3	22	AppliedLearning Literacy_Language	0.148256	0.851744
4	4	19	AppliedLearning Math_Science	0.191489	0.808511
5	5	15	AppliedLearning Music_Arts	0.198238	0.801762
6	6	26	AppliedLearning SpecialNeeds	0.187500	0.812500
7	7	47	AppliedLearning Warmth Care_Hunger	0.400000	0.600000
8	8	7	Health_Sports	0.151746	0.848254
9	9	39	Health_Sports AppliedLearning	0.166667	0.833333
10	10	41	Health_Sports History_Civics	0.062500	0.937500
11	11	27	Health_Sports Literacy_Language	0.187266	0.812734
12	12	17	Health_Sports Math_Science	0.225000	0.775000
13	13	11	Health_Sports Music_Arts	0.219512	0.780488
14	14	24	Health_Sports SpecialNeeds	0.128019	0.871981
15	15	40	Health_Sports Warmth Care_Hunger	0.142857	0.857143
16	16	12	History_Civics	0.181658	0.818342
17	17	45	History_Civics AppliedLearning	0.166667	0.833333
18	18	43	History_Civics Health_Sports	0.078652	1.000000
19	19	20	History_Civics Literacy_Language	0.134454	0.921348
20	20	28	History_Civics Math_Science	0.164706	0.865546
21	21	33	History_Civics Music_Arts	0.246377	0.835294
22	22	23	History_Civics SpecialNeeds	0.135864	0.753623
23	23	6	Literacy_Language	0.159574	0.864136
24	24	38	Literacy_Language AppliedLearning	0.160000	0.840426
25	25	34	Literacy_Language Health_Sports	0.105042	0.840000
26	26	18	Literacy_Language History_Civics	0.135226	0.894958
27	27	8	Literacy_Language Math_Science	0.184411	0.864774
28	28	4	Literacy_Language Music_Arts	0.144781	0.815589
29	29	10	Literacy_Language SpecialNeeds	0.178197	0.855219
30	30	32	Literacy_Language Warmth Care_Hunger	0.149171	1.000000
31	31	2	Math_Science	0.218750	0.821803

	level_0	index	clean_categories	labels	labels
32	32	21	Math_Science AppliedLearning	0.138462	0.850829
33	33	0	Math_Science Health_Sports	0.137881	0.781250
34	34	9	Math_Science History_Civics	0.171937	0.861538
35	35	13	Math_Science Literacy_Language	0.162630	0.862119
36	36	14	Math_Science Music_Arts	0.152830	0.828063
37	37	25	Math_Science SpecialNeeds	0.250000	0.837370
38	38	46	Math_Science Warmth Care_Hunger	0.222222	1.000000
39	39	16	Music_Arts	0.571429	0.847170
40	40	44	Music_Arts AppliedLearning	0.100000	0.750000
41	41	35	Music_Arts Health_Sports	1.000000	0.777778
42	42	42	Music_Arts History_Civics	0.197472	0.428571
43	43	36	Music_Arts SpecialNeeds	0.142857	0.900000
44	44	49	Music_Arts Warmth Care_Hunger	0.215054	0.802528
45	45	5	SpecialNeeds	0.063882	0.857143
46	46	37	SpecialNeeds Health_Sports	NaN	0.784946
47	47	31	SpecialNeeds Music_Arts	NaN	1.000000
48	48	48	SpecialNeeds Warmth Care_Hunger	NaN	0.936118
49	49	3	Warmth Care_Hunger	NaN	NaN

In [29]:
```
category_1.drop(['level_0','index'],axis=1,inplace=True)
category_1.head(2)
```

Out[29]:

	clean_categories	labels	labels
0	AppliedLearning	0.199827	0.800173
1	AppliedLearning Health_Sports	0.164894	0.835106

In [30]:
```
category_1 = df_column_uniquify(category_1)
```

In [31]:
```
category_1.head(2)
```

Out[31]:

	clean_categories	labels	labels_1
0	AppliedLearning	0.199827	0.800173
1	AppliedLearning Health_Sports	0.164894	0.835106

In [32]:
```
category_1.rename(columns={'labels':'Category_0','labels_1':'Category_1'},inpl
ace=True)
```

In [33]:
```
category_1["Category_0"].fillna(value=0, inplace = True)
category_1["Category_1"].fillna(value=0, inplace = True)
```

In [34]:
```
category_1
```

Out[34]:

	clean_categories	Category_0	Category_1
0	AppliedLearning	0.199827	0.800173
1	AppliedLearning Health_Sports	0.164894	0.835106
2	AppliedLearning History_Civics	0.220000	0.780000
3	AppliedLearning Literacy_Language	0.148256	0.851744
4	AppliedLearning Math_Science	0.191489	0.808511
5	AppliedLearning Music_Arts	0.198238	0.801762
6	AppliedLearning SpecialNeeds	0.187500	0.812500
7	AppliedLearning Warmth Care_Hunger	0.400000	0.600000
8	Health_Sports	0.151746	0.848254
9	Health_Sports AppliedLearning	0.166667	0.833333
10	Health_Sports History_Civics	0.062500	0.937500
11	Health_Sports Literacy_Language	0.187266	0.812734
12	Health_Sports Math_Science	0.225000	0.775000
13	Health_Sports Music_Arts	0.219512	0.780488
14	Health_Sports SpecialNeeds	0.128019	0.871981
15	Health_Sports Warmth Care_Hunger	0.142857	0.857143
16	History_Civics	0.181658	0.818342
17	History_Civics AppliedLearning	0.166667	0.833333
18	History_Civics Health_Sports	0.078652	1.000000
19	History_Civics Literacy_Language	0.134454	0.921348
20	History_Civics Math_Science	0.164706	0.865546
21	History_Civics Music_Arts	0.246377	0.835294
22	History_Civics SpecialNeeds	0.135864	0.753623
23	Literacy_Language	0.159574	0.864136
24	Literacy_Language AppliedLearning	0.160000	0.840426
25	Literacy_Language Health_Sports	0.105042	0.840000
26	Literacy_Language History_Civics	0.135226	0.894958
27	Literacy_Language Math_Science	0.184411	0.864774
28	Literacy_Language Music_Arts	0.144781	0.815589
29	Literacy_Language SpecialNeeds	0.178197	0.855219
30	Literacy_Language Warmth Care_Hunger	0.149171	1.000000
31	Math_Science	0.218750	0.821803

	clean_categories	Category_0	Category_1
32	Math_Science AppliedLearning	0.138462	0.850829
33	Math_Science Health_Sports	0.137881	0.781250
34	Math_Science History_Civics	0.171937	0.861538
35	Math_Science Literacy_Language	0.162630	0.862119
36	Math_Science Music_Arts	0.152830	0.828063
37	Math_Science SpecialNeeds	0.250000	0.837370
38	Math_Science Warmth Care_Hunger	0.222222	1.000000
39	Music_Arts	0.571429	0.847170
40	Music_Arts AppliedLearning	0.100000	0.750000
41	Music_Arts Health_Sports	1.000000	0.777778
42	Music_Arts History_Civics	0.197472	0.428571
43	Music_Arts SpecialNeeds	0.142857	0.900000
44	Music_Arts Warmth Care_Hunger	0.215054	0.802528
45	SpecialNeeds	0.063882	0.857143
46	SpecialNeeds Health_Sports	0.000000	0.784946
47	SpecialNeeds Music_Arts	0.000000	1.000000
48	SpecialNeeds Warmth Care_Hunger	0.000000	0.936118
49	Warmth Care_Hunger	0.000000	0.000000

In [35]:
```
project_data_train = pd.merge(project_data_train, category_1, on='clean_catego
ries', how='left').reset_index()
```

In [36]:
```
project_data_train.drop(['index','clean_categories'],axis=1, inplace=True)
```

In [37]:
```
project_data_train.head(1)
```

Out[37]:

	id	teacher_id	teacher_prefix	school_state	project_su
0	p048793	f2bbd023e6288ad1d47cca9db60b34b9	Mrs	NC	8/15/2016

preprocessing of `project_subject_categories` - Test Data

In [38]:
```
catogories = list(project_data_test['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
 space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

In [39]:
```
unique_list_test = []
for x in cat_list:
    if x not in unique_list_test:
            unique_list_test.append(x)

#https://stackoverflow.com/questions/41125909/python-find-elements-in-one-list
-that-are-not-in-the-other

difference=list(set(unique_list_test).difference(unique_list))
print(difference)
```

```
[]
```

In [40]:
```
#df1=pd.DataFrame([['Music_Arts Warmth Care_Hunger',0.5,0.5]],columns=['clean_
categories','Category_0','Category_1'])
```

In [41]:
```
#category_1=category_1.append(df1, ignore_index = True)
```

In [42]:
```
project_data_test = pd.merge(project_data_test, category_1, on='clean_categori
es', how='left').reset_index()
```

In [43]:
```
project_data_test.drop(['clean_categories','Unnamed: 0'],axis=1, inplace=True)
```

preprocessing of project_subject_subcategories - Train Data

In [44]:
```
sub_catogories = list(project_data_train['project_subject_subcategories'].valu
es)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
 space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data_train['clean_subcategories'] = sub_cat_list
project_data_train.drop(['project_subject_subcategories'], axis=1, inplace=Tru
e)
```

In [45]:
```
unique_list = []
for x in sub_cat_list:
    if x not in unique_list:
            unique_list.append(x)


categories=pd.DataFrame({'clean_subcategories': unique_list})
categories=categories.sort_values(['clean_subcategories'], ascending=True).res
et_index()
```

In [46]:
```
df1=project_data_train[['clean_subcategories','labels']][(project_data_train[
'labels']==1)]
```

In [47]:
```
df2=project_data_train[['clean_subcategories','labels']][(project_data_train[
'labels']==0)]
```

In [48]:
```
z =df1.groupby(['clean_subcategories'])['labels'].value_counts() /project_data
_train.groupby(['clean_subcategories'])['labels'].count()
group_1=pd.DataFrame(z)
group_1=group_1.reset_index(drop=True)
```

In [49]:
```
z1 =df2.groupby(['clean_subcategories'])['labels'].value_counts() /project_dat
a_train.groupby(['clean_subcategories'])['labels'].count()
group_0=pd.DataFrame(z1)
group_0=group_0.reset_index(drop=True)
```

In [50]:
```
x1= df1.groupby(['clean_subcategories'])['labels'].value_counts()
class_1=pd.DataFrame(x1)
class_1=class_1.reset_index(drop=True)

x0= df2.groupby(['clean_subcategories'])['labels'].value_counts()
class_0=pd.DataFrame(x0)
class_0=class_0.reset_index(drop=True)
```

In [51]:
```
Response_Table = pd.concat([categories, class_0, class_1],axis=1)
Response_Table = df_column_uniquify(Response_Table)
Response_Table.rename(columns={'labels':'Class=0','labels_1':'Class=1'},inplac
e=True)
print("Response Table for Sub-Categories")
Response_Table
```

Response Table for Sub-Categories

Out[51]:

	index	clean_subcategories	Class=0	Class=1
0	2	AppliedSciences	148.0	586.0
1	202	AppliedSciences CharacterEducation	1.0	11.0
2	237	AppliedSciences Civics_Government	1.0	4.0
3	67	AppliedSciences College_CareerPrep	19.0	111.0
4	278	AppliedSciences CommunityService	1.0	5.0
5	179	AppliedSciences ESL	3.0	28.0
6	42	AppliedSciences EarlyDevelopment	10.0	40.0
7	92	AppliedSciences EnvironmentalScience	63.0	232.0
8	108	AppliedSciences Extracurricular	4.0	40.0
9	355	AppliedSciences FinancialLiteracy	1.0	1.0
10	299	AppliedSciences ForeignLanguages	1.0	3.0
11	198	AppliedSciences Gym_Fitness	29.0	149.0
12	33	AppliedSciences Health_LifeScience	4.0	15.0
13	164	AppliedSciences Health_Wellness	6.0	22.0
14	199	AppliedSciences History_Geography	28.0	152.0
15	76	AppliedSciences Literacy	16.0	104.0
16	22	AppliedSciences Literature_Writing	171.0	854.0
17	34	AppliedSciences Mathematics	2.0	17.0
18	148	AppliedSciences Music	5.0	1.0
19	295	AppliedSciences NutritionEducation	1.0	22.0
20	180	AppliedSciences Other	1.0	20.0
21	97	AppliedSciences ParentInvolvement	1.0	9.0
22	153	AppliedSciences PerformingArts	12.0	16.0
23	18	AppliedSciences SocialSciences	38.0	108.0
24	93	AppliedSciences SpecialNeeds	24.0	3.0
25	339	AppliedSciences TeamSports	5.0	168.0
26	24	AppliedSciences VisualArts	4.0	75.0
27	19	CharacterEducation	16.0	23.0
28	157	CharacterEducation College_CareerPrep	1.0	15.0
29	68	CharacterEducation CommunityService	1.0	4.0
...	...	...	...	...
338	31	Other	NaN	6.0

	index	clean_subcategories	Class=0	Class=1
339	290	Other ParentInvolvement	NaN	4.0
340	356	Other PerformingArts	NaN	2.0
341	345	Other SocialSciences	NaN	1.0
342	58	Other SpecialNeeds	NaN	8.0
343	168	Other TeamSports	NaN	121.0
344	167	Other VisualArts	NaN	6.0
345	318	Other Warmth Care_Hunger	NaN	3.0
346	217	ParentInvolvement	NaN	17.0
347	162	ParentInvolvement PerformingArts	NaN	50.0
348	211	ParentInvolvement SocialSciences	NaN	10.0
349	294	ParentInvolvement SpecialNeeds	NaN	18.0
350	257	ParentInvolvement VisualArts	NaN	1016.0
351	98	PerformingArts	NaN	6.0
352	283	PerformingArts SocialSciences	NaN	73.0
353	152	PerformingArts SpecialNeeds	NaN	4.0
354	141	PerformingArts TeamSports	NaN	266.0
355	74	PerformingArts VisualArts	NaN	1.0
356	145	SocialSciences	NaN	542.0
357	205	SocialSciences SpecialNeeds	NaN	381.0
358	260	SocialSciences VisualArts	NaN	NaN
359	5	SpecialNeeds	NaN	NaN
360	144	SpecialNeeds TeamSports	NaN	NaN
361	120	SpecialNeeds VisualArts	NaN	NaN
362	321	SpecialNeeds Warmth Care_Hunger	NaN	NaN
363	28	TeamSports	NaN	NaN
364	326	TeamSports VisualArts	NaN	NaN
365	26	VisualArts	NaN	NaN
366	346	VisualArts Warmth Care_Hunger	NaN	NaN
367	3	Warmth Care_Hunger	NaN	NaN

368 rows × 4 columns

In [52]:
```
category_1 = pd.concat([categories,group_0,group_1],axis=1).reset_index()
category_1.head(2)
```

Out[52]:

	level_0	index	clean_subcategories	labels	labels
0	0	2	AppliedSciences	0.201635	0.798365
1	1	202	AppliedSciences CharacterEducation	0.083333	0.916667

In [53]:
```
category_1.drop(['index'],axis=1,inplace=True)
category_1.head(2)
```

Out[53]:

	level_0	clean_subcategories	labels	labels
0	0	AppliedSciences	0.201635	0.798365
1	1	AppliedSciences CharacterEducation	0.083333	0.916667

In [54]:
```
category_1 = df_column_uniquify(category_1)
category_1.head(2)
category_1.rename(columns={'labels':'SubCategory_0','labels_1':'SubCategory_1'
},inplace=True)
category_1.head(2)
```

Out[54]:

	level_0	clean_subcategories	SubCategory_0	SubCategory_1
0	0	AppliedSciences	0.201635	0.798365
1	1	AppliedSciences CharacterEducation	0.083333	0.916667

In [55]:
```
category_1["SubCategory_0"].fillna( value=0, inplace = True)
category_1["SubCategory_1"].fillna( value=0, inplace = True)
```

In [56]:
```
project_data_train = pd.merge(project_data_train, category_1, on='clean_subcat
egories', how='left').reset_index()
```

In [57]:
```
project_data_train.drop(['clean_subcategories'],axis=1, inplace=True)
```

preprocessing of `project_subject_subcategories` - Test Data

In [58]:
```
sub_catogories = list(project_data_test['project_subject_subcategories'].value
s)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
 space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

In [59]:
```
unique_list_test = []
for x in sub_cat_list:
    if x not in unique_list_test:
            unique_list_test.append(x)

#https://stackoverflow.com/questions/41125909/python-find-elements-in-one-list
-that-are-not-in-the-other

difference=list(set(unique_list_test).difference(unique_list))
print(difference)
```

```
['Civics_Government PerformingArts', 'AppliedSciences Warmth Care_Hunger', 'C
haracterEducation FinancialLiteracy', 'ParentInvolvement Warmth Care_Hunger',
'ParentInvolvement TeamSports', 'CharacterEducation Civics_Government', 'Gym_
Fitness SocialSciences', 'AppliedSciences Economics', 'Civics_Government ES
L', 'Extracurricular ForeignLanguages', 'ForeignLanguages PerformingArts', 'C
ollege_CareerPrep Warmth Care_Hunger', 'Civics_Government Extracurricular',
'CommunityService FinancialLiteracy', 'Economics Music', 'CommunityService Mu
sic']
```

In [60]:
```
df1=pd.DataFrame([['Civics_Government PerformingArts',0.5,0.5],['AppliedScienc
es Warmth Care_Hunger',0.5,0.5],['CharacterEducation FinancialLiteracy',0.5,0.
5],['ParentInvolvement Warmth Care_Hunger',0.5,0.5],['ParentInvolvement TeamSp
orts',0.5,0.5],['CharacterEducation Civics_Government',0.5,0.5],['Gym_Fitness
 SocialSciences',0.5,0.5],['AppliedSciences Economics',0.5,0.5],['Civics_Gover
nment ESL',0.5,0.5],['Extracurricular ForeignLanguages',0.5,0.5],['ForeignLang
uages PerformingArts',0.5,0.5],['College_CareerPrep Warmth Care_Hunger',0.5,0.
5],['Civics_Government Extracurricular',0.5,0.5],['CommunityService FinancialL
iteracy',0.5,0.5],['Economics Music',0.5,0.5],['CommunityService Music',0.5,0.
5]],columns=['clean_subcategories','SubCategory_0','SubCategory_1'])
```

In [61]:
```
category_1=category_1.append(df1, ignore_index = True)
```

In [62]:
```
category_1.drop(['level_0'],axis=1,inplace=True)
category_1.head(1)
```

Out[62]:

	SubCategory_0	SubCategory_1	clean_subcategories
0	0.201635	0.798365	AppliedSciences

In [63]:
```
project_data_test = pd.merge(project_data_test, category_1, on='clean_subcateg
ories', how='left')
```

Text preprocessing - Train Data

In [64]:
```
# merge two column text dataframe:
project_data_train["essay"] = project_data_train["project_essay_1"].map(str) +
\
                              project_data_train["project_essay_2"].map(str) +
 \
                              project_data_train["project_essay_3"].map(str) +
 \
                              project_data_train["project_essay_4"].map(str)
```

In [65]:
```
# printing some random reviews
print(project_data_train['essay'].values[0])
print("="*50)
```

```
Benjamin Franklin once said, \"Tell me and I forget  Teach me and remember
Involve me and I learn \"  Students need to be involved in their learning so
that they can take ownership of what they are doing  Students need to get the
ir hands \"dirty\" by using them to research, create and make learning meanin
gful  I will teach about 22 eager, energetic 3rd grade students in my classro
om this year   We are a Title I school with 100% participation in the reduced
price/free lunch program  Many of our students have never left the community
nor experienced anything outside of the area  \r\n\r\nAll of our students are
eager to learn and need experiences that they do not get at home  We do not h
ave much parental involvement at our school and we are looking to change that
  We want parents to associate school with love, learning and caring  We woul
d like to change parents' points-of-view from the negative experiences they m
ay have had as a child, to be able to see the positive experiences we provide
their children \r\nThis year our school is starting a greenhouse and a garden
   This idea was brought up by the students! They are thrilled to be able to
have the opportunity to be a part of something so important   \r\nThe student
s want to take responsibility for the garden and be able to actively work in
it throughout the school year   To do this, The students have decided they ne
ed garden tools to be able to plant and weed  The students decided they would
like an irrigation system to make watering the plants more effective   To sto
re all of the tools for the garden and green house, students would like to ke
ep their tools on a workbench  \r\nMy students also want to make sure they ar
e taking care of the garden properly   They would like sets of books to be ab
le to read and research various topics about growing vegetables and other pla
nts  We can't wait to get our hands dirty!nannan
==================================================
```

In [66]:
```
# 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 [67]:
```
sent = decontracted(project_data_train['essay'].values[0])
print(sent)
print("="*50)
```

```
Benjamin Franklin once said, \"Tell me and I forget  Teach me and remember
Involve me and I learn \"  Students need to be involved in their learning so
that they can take ownership of what they are doing  Students need to get the
ir hands \"dirty\" by using them to research, create and make learning meanin
gful  I will teach about 22 eager, energetic 3rd grade students in my classro
om this year   We are a Title I school with 100% participation in the reduced
price/free lunch program  Many of our students have never left the community
nor experienced anything outside of the area  \r\n\r\nAll of our students are
eager to learn and need experiences that they do not get at home  We do not h
ave much parental involvement at our school and we are looking to change that
  We want parents to associate school with love, learning and caring  We woul
d like to change parents' points-of-view from the negative experiences they m
ay have had as a child, to be able to see the positive experiences we provide
their children \r\nThis year our school is starting a greenhouse and a garden
   This idea was brought up by the students! They are thrilled to be able to
have the opportunity to be a part of something so important   \r\nThe student
s want to take responsibility for the garden and be able to actively work in
it throughout the school year   To do this, The students have decided they ne
ed garden tools to be able to plant and weed  The students decided they would
like an irrigation system to make watering the plants more effective   To sto
re all of the tools for the garden and green house, students would like to ke
ep their tools on a workbench  \r\nMy students also want to make sure they ar
e taking care of the garden properly   They would like sets of books to be ab
le to read and research various topics about growing vegetables and other pla
nts  We can not wait to get our hands dirty!nannan
==================================================
```

In [68]:
```
# \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)
```

```
Benjamin Franklin once said,  Tell me and I forget  Teach me and remember   I
nvolve me and I learn    Students need to be involved in their learning so th
at they can take ownership of what they are doing  Students need to get their
hands  dirty  by using them to research, create and make learning meaningful
I will teach about 22 eager, energetic 3rd grade students in my classroom thi
s year   We are a Title I school with 100% participation in the reduced pric
e/free lunch program  Many of our students have never left the community nor
experienced anything outside of the area      All of our students are eager t
o learn and need experiences that they do not get at home  We do not have muc
h parental involvement at our school and we are looking to change that  We wa
nt parents to associate school with love, learning and caring  We would like
to change parents' points-of-view from the negative experiences they may have
had as a child, to be able to see the positive experiences we provide their c
hildren   This year our school is starting a greenhouse and a garden   This i
dea was brought up by the students! They are thrilled to be able to have the
opportunity to be a part of something so important     The students want to t
ake responsibility for the garden and be able to actively work in it througho
ut the school year   To do this, The students have decided they need garden t
ools to be able to plant and weed  The students decided they would like an ir
rigation system to make watering the plants more effective   To store all of
the tools for the garden and green house, students would like to keep their t
ools on a workbench    My students also want to make sure they are taking car
e of the garden properly   They would like sets of books to be able to read a
nd research various topics about growing vegetables and other plants  We can
not wait to get our hands dirty!nannan
```

In [69]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
Benjamin Franklin once said Tell me and I forget Teach me and remember Involv
e me and I learn Students need to be involved in their learning so that they
can take ownership of what they are doing Students need to get their hands di
rty by using them to research create and make learning meaningful I will teac
h about 22 eager energetic 3rd grade students in my classroom this year We ar
e a Title I school with 100 participation in the reduced price free lunch pro
gram Many of our students have never left the community nor experienced anyth
ing outside of the area All of our students are eager to learn and need exper
iences that they do not get at home We do not have much parental involvement
at our school and we are looking to change that We want parents to associate
school with love learning and caring We would like to change parents points o
f view from the negative experiences they may have had as a child to be able
to see the positive experiences we provide their children This year our schoo
l is starting a greenhouse and a garden This idea was brought up by the stude
nts They are thrilled to be able to have the opportunity to be a part of some
thing so important The students want to take responsibility for the garden an
d be able to actively work in it throughout the school year To do this The st
udents have decided they need garden tools to be able to plant and weed The s
tudents decided they would like an irrigation system to make watering the pla
nts more effective To store all of the tools for the garden and green house s
tudents would like to keep their tools on a workbench My students also want t
o make sure they are taking care of the garden properly They would like sets
of books to be able to read and research various topics about growing vegetab
les and other plants We can not wait to get our hands dirty nannan
```

In [70]:
```
# 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', 'it
self', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becau
se', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'a
ll', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
"didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'm
a', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [71]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data_train['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays.append(sent.lower().strip())

# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
my_counter = Counter()
for word in preprocessed_essays:
    my_counter.update(word.split())

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

```
100%|████████████████████████████████████████████████████████████████
██| 33500/33500 [00:24<00:00, 1355.94it/s]
```

In [72]:
```
preprocessed_essays[0]
```

Out[72]:
```
'benjamin franklin said tell forget teach remember involve learn students nee
d involved learning take ownership students need get hands dirty using resear
ch create make learning meaningful teach 22 eager energetic 3rd grade student
s classroom year title school 100 participation reduced price free lunch prog
ram many students never left community nor experienced anything outside area
students eager learn need experiences not get home not much parental involvem
ent school looking change want parents associate school love learning caring
would like change parents points view negative experiences may child able see
positive experiences provide children year school starting greenhouse garden
idea brought students thrilled able opportunity part something important stud
ents want take responsibility garden able actively work throughout school yea
r students decided need garden tools able plant weed students decided would l
ike irrigation system make watering plants effective store tools garden green
house students would like keep tools workbench students also want make sure t
aking care garden properly would like sets books able read research various t
opics growing vegetables plants not wait get hands dirty nannan'
```

Text Preprocessing - For Test Data

In [73]:
```
# merge two column text dataframe:
project_data_test["essay"] = project_data_test["project_essay_1"].map(str) +\
                             project_data_test["project_essay_2"].map(str) + \
                             project_data_test["project_essay_3"].map(str) + \
                             project_data_test["project_essay_4"].map(str)
```

In [74]:
```
# printing some random essays.
print(project_data_test['essay'].values[0])
```

```
The students I have had the privilege of working with are some of the most cr
eative, innovative, and talented kids I know!  Their ability to take any give
n assignment and take it to a new level of understanding never ceases to amaz
e me   I struggle when the resources of our classroom limit their ability to
show their true potential   \r\n\r\nMy previous classes have been right aroun
d 30 students which creates a tight learning space   It is vital I stay organ
ized, well planned, and do my best to create a safe and comfortable space for
my students to be themselves and make mistakes \r\n\r\nParents and the surrou
nding community are very involved in my school   My students seem to thrive o
n the ability to work with others and create ideas that impacts kids outside
of our school walls This project involves a pencil sharpener as well as class
set of white boards   Together these are items my students need to make the m
ost of their learning   \r\n\r\nMy students do so much work together as teams
that these white boards will provide them the opportunity to work out their i
ndividual ideas on their own board before sharing with the group   They can e
asily erase and redo work as they go and the larger white boards allow them t
he space they need to work through challenging math problems as they need to
  \r\n\r\nThis particular pencil sharpener is a classroom staple!  I have had
many sharpeners in my previous years of teaching but only this brand/model se
ems to be able to withstand the hardships of daily use by 30 students   Witho
ut it sharpening pencils is a daily struggle nannan
```

In [75]:
```
# 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 [76]:
```
sent_test = decontracted(project_data_test['essay'].values[0])
print(sent_test)
```

```
The students I have had the privilege of working with are some of the most cr
eative, innovative, and talented kids I know!  Their ability to take any give
n assignment and take it to a new level of understanding never ceases to amaz
e me   I struggle when the resources of our classroom limit their ability to
show their true potential   \r\n\r\nMy previous classes have been right aroun
d 30 students which creates a tight learning space   It is vital I stay organ
ized, well planned, and do my best to create a safe and comfortable space for
my students to be themselves and make mistakes \r\n\r\nParents and the surrou
nding community are very involved in my school   My students seem to thrive o
n the ability to work with others and create ideas that impacts kids outside
of our school walls This project involves a pencil sharpener as well as class
set of white boards   Together these are items my students need to make the m
ost of their learning   \r\n\r\nMy students do so much work together as teams
that these white boards will provide them the opportunity to work out their i
ndividual ideas on their own board before sharing with the group   They can e
asily erase and redo work as they go and the larger white boards allow them t
he space they need to work through challenging math problems as they need to
  \r\n\r\nThis particular pencil sharpener is a classroom staple!  I have had
many sharpeners in my previous years of teaching but only this brand/model se
ems to be able to withstand the hardships of daily use by 30 students   Witho
ut it sharpening pencils is a daily struggle nannan
```

In [77]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-
breaks-python/
sent_test = sent_test.replace('\\r', ' ')
sent_test = sent_test.replace('\\"', ' ')
sent_test = sent_test.replace('\\n', ' ')
print(sent_test)
```

```
The students I have had the privilege of working with are some of the most cr
eative, innovative, and talented kids I know!  Their ability to take any give
n assignment and take it to a new level of understanding never ceases to amaz
e me   I struggle when the resources of our classroom limit their ability to
show their true potential       My previous classes have been right around 30
students which creates a tight learning space   It is vital I stay organized,
well planned, and do my best to create a safe and comfortable space for my st
udents to be themselves and make mistakes     Parents and the surrounding com
munity are very involved in my school   My students seem to thrive on the abi
lity to work with others and create ideas that impacts kids outside of our sc
hool walls This project involves a pencil sharpener as well as class set of w
hite boards   Together these are items my students need to make the most of t
heir learning       My students do so much work together as teams that these
white boards will provide them the opportunity to work out their individual i
deas on their own board before sharing with the group   They can easily erase
and redo work as they go and the larger white boards allow them the space the
y need to work through challenging math problems as they need to       This p
articular pencil sharpener is a classroom staple!  I have had many sharpeners
in my previous years of teaching but only this brand/model seems to be able t
o withstand the hardships of daily use by 30 students   Without it sharpening
pencils is a daily struggle nannan
```

In [78]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_test = re.sub('[^A-Za-z0-9]+', ' ', sent_test)
print(sent_test)
```

```
The students I have had the privilege of working with are some of the most cr
eative innovative and talented kids I know Their ability to take any given as
signment and take it to a new level of understanding never ceases to amaze me
I struggle when the resources of our classroom limit their ability to show th
eir true potential My previous classes have been right around 30 students whi
ch creates a tight learning space It is vital I stay organized well planned a
nd do my best to create a safe and comfortable space for my students to be th
emselves and make mistakes Parents and the surrounding community are very inv
olved in my school My students seem to thrive on the ability to work with oth
ers and create ideas that impacts kids outside of our school walls This proje
ct involves a pencil sharpener as well as class set of white boards Together
these are items my students need to make the most of their learning My studen
ts do so much work together as teams that these white boards will provide the
m the opportunity to work out their individual ideas on their own board befor
e sharing with the group They can easily erase and redo work as they go and t
he larger white boards allow them the space they need to work through challen
ging math problems as they need to This particular pencil sharpener is a clas
sroom staple I have had many sharpeners in my previous years of teaching but
only this brand model seems to be able to withstand the hardships of daily us
e by 30 students Without it sharpening pencils is a daily struggle nannan
```

In [79]:
```
# 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', 'it
self', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becau
se', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'a
ll', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
"didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'm
a', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [80]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays_test = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data_test['essay'].values):
    sent_cv = decontracted(sentence)
    sent_cv = sent_cv.replace('\\r', ' ')
    sent_cv = sent_cv.replace('\\"', ' ')
    sent_cv = sent_cv.replace('\\n', ' ')
    sent_cv = re.sub('[^A-Za-z0-9]+', ' ', sent_cv)
    # https://gist.github.com/sebleier/554280
    sent_cv = ' '.join(e for e in sent_cv.split() if e not in stopwords)
    preprocessed_essays_test.append(sent_cv.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████
██| 16500/16500 [00:10<00:00, 1540.56it/s]
```

In [81]:
```
# after preprocesing
preprocessed_essays_test[0]
```

Out[81]:
```
'the students i privilege working creative innovative talented kids i know th
eir ability take given assignment take new level understanding never ceases a
maze i struggle resources classroom limit ability show true potential my prev
ious classes right around 30 students creates tight learning space it vital i
stay organized well planned best create safe comfortable space students make
mistakes parents surrounding community involved school my students seem thriv
e ability work others create ideas impacts kids outside school walls this pro
ject involves pencil sharpener well class set white boards together items stu
dents need make learning my students much work together teams white boards pr
ovide opportunity work individual ideas board sharing group they easily erase
redo work go larger white boards allow space need work challenging math probl
ems need this particular pencil sharpener classroom staple i many sharpeners
previous years teaching brand model seems able withstand hardships daily use
30 students without sharpening pencils daily struggle nannan'
```

Preprocessing of `project_title` - Train Data

In [82]:
```
# printing some random title.
print(project_data_train['project_title'].values[0])
print("="*50)
```

```
Let's Get Growing!
==================================================
```

In [83]:
```
sent = decontracted(project_data_train['project_title'].values[0])
print(sent)
print("="*50)
```

```
Let is Get Growing!
==================================================
```

In [84]:
```
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
Let is Get Growing!
```

In [85]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data_train['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_title.append(sent.lower().strip())

my_counter = Counter()
for word in preprocessed_title:
    my_counter.update(word.split())

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

```
100%|████████████████████████████████████████████████████████████████
| 33500/33500 [00:01<00:00, 30561.61it/s]
```

In [86]:
```
# after preprocesing
preprocessed_title[0]
```

Out[86]: 'let get growing'

Preprocessing of `project_title - For Test Data`

In [87]:
```
# printing some random title.
print(project_data_test['project_title'].values[0])
print("="*50)
```

```
Dry Erase Boards ? A safe place for mistakes!
==================================================
```

In [88]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_title_test = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data_test['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_title_test.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████
| 16500/16500 [00:00<00:00, 32503.19it/s]
```

In [89]:
```
# after preprocesing
preprocessed_title_test[0]
```

Out[89]:
```
'dry erase boards a safe place mistakes'
```

1.5 Preparing data for models

In [90]:
```
project_data_train.columns
```

Out[90]:
```
Index(['index', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'labels', 'Category_
0',
       'Category_1', 'level_0', 'SubCategory_0', 'SubCategory_1', 'essay'],
      dtype='object')
```

In [91]:
```
project_data_test.columns
```

Out[91]:
```
Index(['index', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'Category_0',
       'Category_1', 'clean_subcategories', 'SubCategory_0', 'SubCategory_1',
       'essay'],
      dtype='object')
```

Encoding for State - Train Data

In [92]:
```
state = list(project_data_train['school_state'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

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

state_list = []
for i in state:
    temp = ""

    for j in i.split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    state_list.append(temp.strip())

project_data_train['clean_state'] = state_list
project_data_train.drop(['school_state'], axis=1, inplace=True)
```

In [93]:
```
unique_list = []
for x in state_list:
    if x not in unique_list:
            unique_list.append(x)


categories=pd.DataFrame({'clean_state': unique_list})
categories=categories.sort_values(['clean_state'], ascending=True).reset_index
()

df1=project_data_train[['clean_state','labels']][(project_data_train['labels']
==1)]

df2=project_data_train[['clean_state','labels']][(project_data_train['labels']
==0)]
```

In [94]:
```
z =df1.groupby(['clean_state'])['labels'].value_counts() /project_data_train.g
roupby(['clean_state'])['labels'].count()

group_1=pd.DataFrame(z)

group_1=group_1.reset_index(drop=True)
print(group_1.head(2))
```

```
     labels
0  0.830189
1  0.864151
```

In [95]:
```
z1 =df2.groupby(['clean_state'])['labels'].value_counts() /project_data_train.
groupby(['clean_state'])['labels'].count()

group_0=pd.DataFrame(z1)

group_0=group_0.reset_index(drop=True)
print(group_0.head(2))
```

```
     labels
0  0.169811
1  0.135849
```

In [96]:
```
x1= df1.groupby(['clean_state'])['labels'].value_counts()
class_1=pd.DataFrame(x1)
class_1=class_1.reset_index(drop=True)
print ( class_1.head(2))
```

```
   labels
0      88
1     458
```

In [97]:
```
x0= df2.groupby(['clean_state'])['labels'].value_counts()
class_0=pd.DataFrame(x0)
class_0=class_0.reset_index(drop=True)
print ( class_0.head(2))
```

```
   labels
0      18
1      72
```

In [98]:
```
Response_Table = pd.concat([categories, class_0, class_1],axis=1)
Response_Table = df_column_uniquify(Response_Table)
Response_Table.rename(columns={'labels':'Class=0','labels_1':'Class=1'},inplac
e=True)

print("Response Table for State")
Response_Table
```

Response Table for State

Out[98]:

	index	clean_state	Class=0	Class=1
0	40	AK	18	88
1	25	AL	72	458
2	1	AR	53	248
3	26	AZ	112	564
4	7	CA	688	3974
5	17	CO	66	307
6	34	CT	68	451
7	46	DC	33	132
8	44	DE	10	95
9	6	FL	327	1592
10	5	GA	205	1022
11	29	HI	22	150
12	8	IA	30	170
13	18	ID	41	154
14	3	IL	200	1148
15	14	IN	125	668
16	48	KS	24	142
17	19	KY	59	367
18	30	LA	125	614
19	28	MA	104	607
20	39	MD	70	384
21	35	ME	24	117
22	24	MI	148	838
23	15	MN	57	345
24	4	MO	120	669
25	2	MS	67	331
26	41	MT	19	52
27	0	NC	227	1297
28	50	ND	5	39
29	42	NE	21	73
30	47	NH	12	80
31	11	NJ	113	558

	index	clean_state	Class=0	Class=1
32	33	NM	21	137
33	32	NV	68	400
34	10	NY	315	1985
35	20	OH	91	680
36	22	OK	120	606
37	37	OR	67	317
38	12	PA	134	821
39	45	RI	17	64
40	9	SC	184	1058
41	38	SD	14	81
42	31	TN	77	437
43	16	TX	425	1764
44	21	UT	90	431
45	27	VA	92	531
46	49	VT	4	17
47	23	WA	74	645
48	13	WI	87	464
49	36	WV	19	128
50	43	WY	4	32

In [99]:
```
category_1 = pd.concat([categories,group_0,group_1],axis=1).reset_index()
category_1.head(2)
```

Out[99]:

	level_0	index	clean_state	labels	labels
0	0	40	AK	0.169811	0.830189
1	1	25	AL	0.135849	0.864151

In [100]:
```
category_1.drop(['level_0','index'],axis=1,inplace=True)
print("Response Table For Categories")
category_1.head(2)
```

```
Response Table For Categories
```

Out[100]:

	clean_state	labels	labels
0	AK	0.169811	0.830189
1	AL	0.135849	0.864151

In [101]:
```
category_1 = df_column_uniquify(category_1)
category_1.head(2)
```

Out[101]:

	clean_state	labels	labels_1
0	AK	0.169811	0.830189
1	AL	0.135849	0.864151

In [102]:
```
category_1.rename(columns={'labels':'State_0','labels_1':'State_1'},inplace=True)
category_1.head(2)
```

Out[102]:

	clean_state	State_0	State_1
0	AK	0.169811	0.830189
1	AL	0.135849	0.864151

In [103]:
```
#category_1["State_0"].fillna( method ='ffill', inplace = True)
#category_1["State_1"].fillna( method ='ffill', inplace = True)
```

In [104]:
```
project_data_train = pd.merge(project_data_train, category_1, on='clean_state', how='left')
project_data_train.drop(['clean_state'],axis=1, inplace=True)
```

In [105]:
```
Cat_0 = list(project_data_train['State_0'].values)
Category_Class_0 = []
for i in Cat_0:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("NaN",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    Category_Class_0.append(temp.strip())

project_data_train['State_Class_0'] = Category_Class_0
project_data_train.drop(['State_0'], axis=1, inplace=True)
```

In [106]:
```
Cat_1 = list(project_data_train['State_1'].values)
Category_Class_1 = []
for i in Cat_1:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("NaN",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    Category_Class_1.append(temp.strip())

project_data_train['State_Class_1'] = Category_Class_1
project_data_train.drop(['State_1'], axis=1, inplace=True)
```

Encoding for State- Test Data

In [107]:
```
state = list(project_data_test['school_state'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

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

state_list_test = []
for i in state:
    temp = ""

    for j in i.split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    state_list_test.append(temp.strip())
```

In [108]:
```
project_data_test['clean_state'] = state_list_test
project_data_test.drop(['school_state'], axis=1, inplace=True)
```

In [109]:
```
unique_list_test = []
for x in state_list_test:
    if x not in unique_list_test:
            unique_list_test.append(x)

#https://stackoverflow.com/questions/41125909/python-find-elements-in-one-list
-that-are-not-in-the-other

difference=list(set(unique_list_test).difference(unique_list))
print(difference)
```

```
[]
```

In [110]:
```
project_data_test = pd.merge(project_data_test, category_1, on='clean_state',
how='left')
project_data_test.drop(['clean_state'],axis=1, inplace=True)
```

In [111]:
```
State_0 = list(project_data_test['State_0'].values)

State_Class_0 = []
for i in State_0:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("nan",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    State_Class_0.append(temp.strip())

project_data_test['State_Class_0'] = State_Class_0
project_data_test.drop(['State_0'], axis=1, inplace=True)
```

In [112]:
```
State_1 = list(project_data_test['State_1'].values)

State_Class_1 = []
for i in State_1:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("nan",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    State_Class_1.append(temp.strip())

project_data_test['State_Class_1'] = State_Class_1
project_data_test.drop(['State_1'], axis=1, inplace=True)
```

Hot Encoding Project Grade Category - Train Data

In [113]:
```
grade = list(project_data_train['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

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

grade_list = []
for i in grade:
    temp = ""

    for j in i.split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("nan",'')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    grade_list.append(temp.strip())

project_data_train['clean_grade'] = grade_list
project_data_train.drop(['project_grade_category'], axis=1, inplace=True)
```

In [114]:
```
unique_list = []
for x in grade_list:
    if x not in unique_list:
            unique_list.append(x)


categories=pd.DataFrame({'clean_grade': unique_list})
categories=categories.sort_values(['clean_grade'], ascending=True).reset_index
()
```

In [115]:
```
df1=project_data_train[['clean_grade','labels']][(project_data_train['labels']
==1)]
print(df1.head(2))
```

```
    clean_grade  labels
0     Grades3-5       1
3  GradesPreK-2       1
```

In [116]:
```
df2=project_data_train[['clean_grade','labels']][(project_data_train['labels']
==0)]
```

In [117]:
```
z =df1.groupby(['clean_grade'])['labels'].value_counts() /project_data_train.g
roupby(['clean_grade'])['labels'].count()

group_1=pd.DataFrame(z)

group_1=group_1.reset_index(drop=True)
print(group_1.head(2))
```

```
     labels
0  0.855133
1  0.836618
```

In [118]:
```
z1 =df2.groupby(['clean_grade'])['labels'].value_counts() /project_data_train.
groupby(['clean_grade'])['labels'].count()

group_0=pd.DataFrame(z1)

group_0=group_0.reset_index(drop=True)
print(group_0.head(2))
```

```
     labels
0  0.144867
1  0.163382
```

In [119]:
```
x1= df1.groupby(['clean_grade'])['labels'].value_counts()
class_1=pd.DataFrame(x1)
class_1=class_1.reset_index(drop=True)
print ( class_1.head(2))
```

```
   labels
0    9787
1    4373
```

In [120]:
```
x0= df2.groupby(['clean_grade'])['labels'].value_counts()
class_0=pd.DataFrame(x0)
class_0=class_0.reset_index(drop=True)
print ( class_0.head(2))
```

```
   labels
0    1658
1     854
```

In [121]:
```
Response_Table = pd.concat([categories, class_0, class_1],axis=1)
Response_Table = df_column_uniquify(Response_Table)
Response_Table.rename(columns={'labels':'Class=0','labels_1':'Class=1'},inplace=True)
Response_Table
```

Out[121]:

	index	clean_grade	Class=0	Class=1
0	0	Grades3-5	1658	9787
1	3	Grades6-8	854	4373
2	2	Grades9-12	556	2733
3	1	GradesPreK-2	2100	11439

In [122]:
```
category_1 = pd.concat([categories,group_0,group_1],axis=1).reset_index()
category_1.head(2)
```

Out[122]:

	level_0	index	clean_grade	labels	labels
0	0	0	Grades3-5	0.144867	0.855133
1	1	3	Grades6-8	0.163382	0.836618

In [123]:
```
category_1.drop(['level_0','index'],axis=1,inplace=True)
print("Response Table For Categories")
category_1.head(2)
```

```
Response Table For Categories
```

Out[123]:

	clean_grade	labels	labels
0	Grades3-5	0.144867	0.855133
1	Grades6-8	0.163382	0.836618

In [124]:
```
category_1 = df_column_uniquify(category_1)

category_1.head(2)

category_1.rename(columns={'labels':'Grade_0','labels_1':'Grade_1'},inplace=True)
category_1.head(2)
```

Out[124]:

	clean_grade	Grade_0	Grade_1
0	Grades3-5	0.144867	0.855133
1	Grades6-8	0.163382	0.836618

In [125]:
```
#category_1["Grade_0"].fillna( method ='ffill', inplace = True)
#category_1["Grade_1"].fillna( method ='ffill', inplace = True)
```

In [126]:
```
project_data_train = pd.merge(project_data_train, category_1, on='clean_grade'
, how='left')
project_data_train.drop(['clean_grade'],axis=1, inplace=True)
```

Hot Encoding Project Grade Category - Test Data

In [127]:
```
grade = list(project_data_test['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

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

grade_list_test = []
for i in grade:
    temp = ""

    for j in i.split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("NaN",'')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    grade_list_test.append(temp.strip())
```

In [128]:
```
project_data_test['clean_grade'] = grade_list_test
project_data_test.drop(['project_grade_category'], axis=1, inplace=True)
```

In [129]:
```
unique_list_test = []
for x in grade_list_test:
    if x not in grade_list_test:
            unique_list_test.append(x)

#https://stackoverflow.com/questions/41125909/python-find-elements-in-one-list
-that-are-not-in-the-other

difference=list(set(unique_list_test).difference(unique_list))
print(difference)
```

```
[]
```

In [130]:
```
project_data_test = pd.merge(project_data_test, category_1, on='clean_grade',
how='left')
```

In [131]:
```
project_data_test.drop(['clean_grade'],axis=1, inplace=True)
```

In [132]:
```
State_0 = list(project_data_test['Grade_0'].values)

State_Class_0 = []
for i in State_0:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("nan",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    State_Class_0.append(temp.strip())

project_data_test['Grade_Class_0'] = State_Class_0
project_data_test.drop(['Grade_0'], axis=1, inplace=True)
```

In [133]:
```
State_1 = list(project_data_test['Grade_1'].values)

State_Class_1 = []
for i in State_1:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("nan",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    State_Class_1.append(temp.strip())

project_data_test['Grade_Class_1'] = State_Class_1
project_data_test.drop(['Grade_1'], axis=1, inplace=True)
```

Hot Encoding Teacher Prefix - Train Data

In [134]:
```
prefix = list(project_data_train['teacher_prefix'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

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

prefix_list = []
for i in prefix:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("nan",'')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    prefix_list.append(temp.strip())
```

In [135]:
```
project_data_train['clean_prefix'] = prefix_list
project_data_train.drop(['teacher_prefix'], axis=1, inplace=True)
project_data_train.head(2)
```

Out[135]:

	index	id	teacher_id	project_submitted_datetime	pro
0	0	p048793	f2bbd023e6288ad1d47cca9db60b34b9	8/15/2016 16:40	Let's Gro
1	1	p007982	9d3c875fc42e2369cd0263b38b9421bb	8/15/2016 16:35	Leg The Bloc Buil You

2 rows × 23 columns

In [136]:
```
unique_list = []
for x in prefix_list:
    if x not in unique_list:
            unique_list.append(x)


categories=pd.DataFrame({'clean_prefix': unique_list})
categories=categories.sort_values(['clean_prefix'], ascending=True).reset_inde
x()
print(categories.head(2))
```

```
   index clean_prefix
0      5
1      4           Dr
```

In [137]:
```
df1=project_data_train[['clean_prefix','labels']][(project_data_train['labels'
]==1)]
df2=project_data_train[['clean_prefix','labels']][(project_data_train['labels'
]==0)]
```

In [138]:
```
z =df1.groupby(['clean_prefix'])['labels'].value_counts() /project_data_train.
groupby(['clean_prefix'])['labels'].count()
group_1=pd.DataFrame(z)
group_1=group_1.reset_index(drop=True)

z1 =df2.groupby(['clean_prefix'])['labels'].value_counts() /project_data_train
.groupby(['clean_prefix'])['labels'].count()
group_0=pd.DataFrame(z1)
group_0=group_0.reset_index(drop=True)
```

In [139]:
```
x1= df1.groupby(['clean_prefix'])['labels'].value_counts()
class_1=pd.DataFrame(x1)
class_1=class_1.reset_index(drop=True)

x0= df2.groupby(['clean_prefix'])['labels'].value_counts()
class_0=pd.DataFrame(x0)
class_0=class_0.reset_index(drop=True)
```

In [140]:
```
Response_Table = pd.concat([categories, class_0, class_1],axis=1)
Response_Table = df_column_uniquify(Response_Table)
Response_Table.rename(columns={'labels':'Class=0','labels_1':'Class=1'},inplace=True)
Response_Table
```

Out[140]:

	index	clean_prefix	Class=0	Class=1
0	5		1.0	2
1	4	Dr	533.0	1
2	2	Mr	2612.0	2726
3	0	Mrs	1890.0	15054
4	1	Ms	132.0	10004
5	3	Teacher	NaN	545

In [141]:
```
category_1 = pd.concat([categories,group_0,group_1],axis=1).reset_index()
category_1.drop(['level_0','index'],axis=1,inplace=True)
```

In [142]:
```
category_1 = df_column_uniquify(category_1)
category_1.rename(columns={'labels':'Prefix_0','labels_1':'Prefix_1'},inplace=True)
```

In [143]:
```
#category_1["Prefix_0"].fillna( value=0, inplace = True)
#category_1["Prefix_1"].fillna( value=0, inplace = True)
```

In [144]:
```
project_data_train = pd.merge(project_data_train, category_1, on='clean_prefix', how='left')
project_data_train.drop(['clean_prefix'],axis=1, inplace=True)
#project_data_train.head(1)
```

In [145]:
```
Cat_0 = list(project_data_train['Prefix_0'].values)
Category_Class_0 = []
for i in Cat_0:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("NaN",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    Category_Class_0.append(temp.strip())

project_data_train['Prefix_Class_0'] = Category_Class_0
project_data_train.drop(['Prefix_0'], axis=1, inplace=True)
```

In [146]:
```
Cat_1 = list(project_data_train['Prefix_1'].values)
Category_Class_1 = []
for i in Cat_1:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("NaN",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    Category_Class_1.append(temp.strip())

project_data_train['Prefix_Class_1'] = Category_Class_1
project_data_train.drop(['Prefix_1'], axis=1, inplace=True)
```

Hot Encoding Teacher Prefix - Test Data

In [147]:
```
prefix = list(project_data_test['teacher_prefix'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

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

prefix_list_test = []
for i in prefix:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("nan",'')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    prefix_list_test.append(temp.strip())
```

In [148]:
```
project_data_test['clean_prefix'] = prefix_list_test
project_data_test.drop(['teacher_prefix'], axis=1, inplace=True)
```

In [149]:
```
unique_list_test = []
for x in prefix_list_test:
    if x not in unique_list_test:
            unique_list_test.append(x)

#https://stackoverflow.com/questions/41125909/python-find-elements-in-one-list
-that-are-not-in-the-other

difference=list(set(unique_list_test).difference(unique_list))
print(difference)
```

```
[]
```

In [150]:
```
#df1=pd.DataFrame([[' '' ' ,0.5,0.5]],columns=['clean_prefix','Prefix_0','Pref
ix_1'])
```

In [151]:
```
#category_1=category_1.append(df1, ignore_index = True)
```

In [152]:
```
project_data_test = pd.merge(project_data_test, category_1, on='clean_prefix',
 how='left')
project_data_test.drop(['clean_prefix'],axis=1, inplace=True)
```

In [153]:
```
Prefix_0 = list(project_data_test['Prefix_0'].values)

Prefix_0 = []
for i in Prefix_0:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("nan",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    Prefix_0.append(temp.strip())

project_data_test['Prefix_Class_0'] = State_Class_0
project_data_test.drop(['Prefix_0'], axis=1, inplace=True)
```

In [154]:
```
Prefix_1 = list(project_data_test['Prefix_1'].values)

Prefix_1 = []
for i in Prefix_1:
    temp = ""

    for j in str(i).split(','): # it will split it in parts
        if 'The' in j.split(): # this will split each of the state based on sp
ace
            j=j.replace('The','') # if we have the words "The" we are going to
 replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty)
        j = j.replace("nan",'0')
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    Prefix_1.append(temp.strip())

project_data_test['Prefix_Class_1'] = State_Class_1
project_data_test.drop(['Prefix_1'], axis=1, inplace=True)
```

Vectorizing Text data

Bag of words - Train Data

In [155]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws or projects).
vectorizer6 = CountVectorizer(min_df=10, lowercase=False, binary=True)
text_bow = vectorizer6.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)

#print(text_bow)
```

```
Shape of matrix after one hot encodig  (33500, 10337)
```

Bag of Words Title - Train Data

In [156]:
```
vectorizer7=CountVectorizer(lowercase=False, binary=True, min_df=0)
title_bow = vectorizer7.fit_transform(preprocessed_title)
print("Shape of matrix after one hot encoding ",title_bow.shape)
```

```
Shape of matrix after one hot encoding  (33500, 9669)
```

Bag of Words Essay- Test Data

In [157]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws or projects).
#vectorizer = CountVectorizer(min_df=10, ngram_range=(2,2), lowercase=False, b
inary=True, max_features=5000, )
text_bow_test = vectorizer6.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encodig ",text_bow_test.shape)
```

```
Shape of matrix after one hot encodig  (16500, 10337)
```

Bag of Words Words Tittle - Test Data

In [158]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws or titles).
#vectorizer = CountVectorizer(vocabulary=list(sorted_title_dict.keys()), lower
case=False, binary=True, min_df=0)
title_bow_test = vectorizer7.transform(preprocessed_title_test)
print("Shape of matrix after one hot encoding ",title_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (16500, 9669)
```

TFIDF vectorizer Essays - Train Data

In [159]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer8 = TfidfVectorizer(min_df=10,lowercase=False, binary=True, max_feat
ures=5000)
text_tfidf = vectorizer8.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (33500, 5000)
```

TFIDF Vectorizer Tittle - Train Data

In [160]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer9 = TfidfVectorizer(min_df=0, lowercase=False, binary=True, max_feat
ures=5000)
tittle_tfidf = vectorizer9.fit_transform(preprocessed_title)
print("Shape of matrix after one hot encoding ",tittle_tfidf.shape)
```

```
Shape of matrix after one hot encoding  (33500, 5000)
```

TFIDF Vectorizer Essay - Test Data

In [161]:
```
#from sklearn.feature_extraction.text import TfidfVectorizer
#vectorizer = TfidfVectorizer(vocabulary=sorted_essays_dict.keys(), lowercase=
False, binary=True, min_df=10)
vectorizer8.fit(preprocessed_essays_test)
text_tfidf_test = vectorizer8.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encodig ",text_tfidf_test.shape)
```

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

TFIDF Vectorizer Tittle - Test Data

In [162]:
```
#from sklearn.feature_extraction.text import TfidfVectorizer
#vectorizer = TfidfVectorizer(vocabulary=list(sorted_title_dict.keys()), lower
case=False, binary=True, min_df=0)
vectorizer9.fit(preprocessed_title_test)
title_tfidf_test = vectorizer9.transform(preprocessed_title_test)
print("Shape of matrix after one hot encodig ",title_tfidf_test.shape)
```

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

1.5.2.3 Using Pretrained Models: Avg W2V

In [163]:
```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

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

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

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

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


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

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

```
Loading Glove Model

1917495it [05:33, 5756.10it/s]

Done. 1917495  words loaded!
all the words in the coupus 4762978
the unique words in the coupus 36723
The number of words that are present in both glove vectors and our coupus 34058 ( 92.743 %)
word 2 vec length 34058
```

In [164]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how
-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

AVG_W2V_Vectors Essays- Train Data -

In [165]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this
 list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

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

```
100%|████████████████████████████████████████████████████████████████████
██| 33500/33500 [00:12<00:00, 2721.81it/s]

33500
300
```

AVG_W2V_Vectors Tittle - Train Data -

In [166]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_tittle = []; # the avg-w2v for each title is stored in this li
st
for sentence in tqdm(preprocessed_title): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_tittle.append(vector)

print(len(avg_w2v_vectors_tittle))
print(len(avg_w2v_vectors_tittle[0]))
```

```
100%|████████████████████████████████████████████████████████████████
█| 33500/33500 [00:00<00:00, 52076.87it/s]

33500
300
```

AVG W2V on Essays- Test Data

In [167]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in
 this list
for sentence in tqdm(preprocessed_essays_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

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

```
100%|████████████████████████████████████████████████████████████████
██| 16500/16500 [00:06<00:00, 2583.12it/s]

16500
300
```

AVG W2V of Tittle Test Data

In [168]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_tittle_test = []; # the avg-w2v for each title is stored in th
is list
for sentence in tqdm(preprocessed_title_test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the title
    for word in sentence.split(): # for each word in a title
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_tittle_test.append(vector)

print(len(avg_w2v_vectors_tittle_test))
print(len(avg_w2v_vectors_tittle_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████
█| 16500/16500 [00:00<00:00, 35051.12it/s]

16500
300
```

Using Pretrained Models: TFIDF weighted W2V for Train Data - Essays

In [169]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model_essays = TfidfVectorizer()
tfidf_model_essays.fit_transform(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_essays.get_feature_names(), list(tfidf_model
_essays.idf_)))
tfidf_words = set(tfidf_model_essays.get_feature_names())
```

In [170]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in th
is list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/revie
w
    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 v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

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

```
100%|████████████████████████████████████████████████████████████████████
███| 33500/33500 [01:25<00:00, 390.07it/s]

33500
300
```

Using Pretrained Models: TFIDF weighted W2V on `project_title` for Train Data

In [171]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model_title = TfidfVectorizer()
tfidf_model_title.fit_transform(preprocessed_title)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_title.get_feature_names(), list(tfidf_model_
title.idf_)))
tfidf_words = set(tfidf_model_title.get_feature_names())
```

In [172]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_Title = []; # the avg-w2v for each sentence/review is stored
 in this list
for sentence in tqdm(preprocessed_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/revie
w
    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 v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_Title.append(vector)

print(len(tfidf_w2v_vectors_Title))
print(len(tfidf_w2v_vectors_Title[0]))
```

```
100%|████████████████████████████████████████████████████████████████████
█| 33500/33500 [00:01<00:00, 22991.44it/s]

33500
300
```

TFIDF weighted W2V Essays for Test Data

In [173]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
#vectorizer = TfidfVectorizer(vocabulary=sorted_essays_dict.keys(), lowercase=
False, binary=True, min_df=10)
tfidf_model_essays.fit(preprocessed_essays_test)
tfidf_model_essays.transform(preprocessed_essays_test)
# we are converting a dictionary with word as a key, and the idf as a value
#dictionary = dict(zip(tfidf_model_essays.get_feature_names(), list(tfidf_mode
l_essays.idf_)))
#tfidf_words = set(tfidf_model_essays.get_feature_names())
```

Out[173]:
```
<16500x27273 sparse matrix of type '<class 'numpy.float64'>'
        with 1783531 stored elements in Compressed Sparse Row format>
```

In [174]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored
 in this list
for sentence in tqdm(preprocessed_essays_test): # 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/revie
w
    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 v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████
███| 16500/16500 [00:45<00:00, 361.04it/s]

16500
300
```

TFIDF Weighted AVG_W2V Title for Test Data

In [175]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
#tfidf_model = TfidfVectorizer(vocabulary=sorted_title_dict.keys(), lowercase=
False, binary=True, min_df=0)
tfidf_model_title.fit(preprocessed_title_test)
tfidf_model_title.transform(preprocessed_title_test)
# we are converting a dictionary with word as a key, and the idf as a value
#dictionary = dict(zip(tfidf_model_title.get_feature_names(), list(tfidf_model
_title.idf_)))
#tfidf_words = set(tfidf_model_title.get_feature_names())
```

Out[175]:
```
<16500x7023 sparse matrix of type '<class 'numpy.float64'>'
        with 68894 stored elements in Compressed Sparse Row format>
```

In [176]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_Title_test = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm(preprocessed_title_test): # 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/revie
w
    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 v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_Title_test.append(vector)

print(len(tfidf_w2v_vectors_Title_test))
print(len(tfidf_w2v_vectors_Title_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████
██| 16500/16500 [00:00<00:00, 24729.66it/s]

16500
300
```

Vectorizing Numerical features for Train Data

In [177]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'
}).reset_index()
project_data_train = pd.merge(project_data_train, price_data, on='id', how='le
ft')
```

In [178]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

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

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

```
Mean : 298.66870089552236, Standard deviation : 383.3080407417817
```

In [179]:
```
price_standardized.shape
```

Out[179]:
```
(33500, 1)
```

Vectorizing Quantity - Train Data

In [180]:
```
import warnings
warnings.filterwarnings("ignore")


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

# Now standardize the data with above maen and variance.
quantity_standardized = quantity_scalar.transform(project_data_train['quantity'].values.reshape(-1, 1))
```

```
Mean : 298.66870089552236, Standard deviation : 383.3080407417817
```

In [181]:
```
quantity_standardized
```

Out[181]:
```
array([[ 0.87780595],
       [-0.18662824],
       [-0.51696988],
       ...,
       [-0.44356063],
       [-0.51696988],
       [ 0.25382729]])
```

Verctorizing for Teacher Previously Posted Projected for Train Data -

In [182]:
```
import warnings
warnings.filterwarnings("ignore")

teacher_number_of_previously_posted_projects_scalar = StandardScaler()
teacher_number_of_previously_posted_projects_scalar.fit_transform(project_data
_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)) #
 finding the mean and standard deviation of this data
print(f"Mean : {teacher_number_of_previously_posted_projects_scalar.mean_[0]},
 Standard deviation : {np.sqrt(teacher_number_of_previously_posted_projects_sc
alar.var_[0])}")

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized = teacher_number_of_
previously_posted_projects_scalar.transform(project_data_train['teacher_number
_of_previously_posted_projects'].values.reshape(-1, 1))
```

```
Mean : 11.169462686567163, Standard deviation : 28.161038481703955
```

In [183]:
```
teacher_number_of_previously_posted_projects_standardized
```

Out[183]:
```
array([[ 0.70418345],
       [-0.32560812],
       [-0.36111817],
       ...,
       [-0.04152768],
       [-0.25458801],
       [-0.21907795]])
```

Vectorizing Numerical features for Test Data

In [184]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'
}).reset_index()
project_data_test = pd.merge(project_data_test, price_data, on='id', how='lef
t')
```

In [185]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

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

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

```
Mean : 300.6837775757576, Standard deviation : 367.6360573746355
```

In [186]:
```
price_standardized_test.shape
```

Out[186]:
```
(16500, 1)
```

Vectorizing Quantity - Test

In [187]:
```
#quantity_scalar = StandardScaler()
quantity_scalar.fit_transform(project_data_test['quantity'].values.reshape(-1, 1)) # finding the mean and standard deviation of this data
print(f"Mean : {quantity_scalar.mean_[0]}, Standard deviation : {np.sqrt(quantity_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
quantity_test_standardized = quantity_scalar.transform(project_data_test['quantity'].values.reshape(-1, 1))
```

```
Mean : 16.99921212121212, Standard deviation : 25.89780497280706
```

In [188]:
```
quantity_test_standardized.shape
```

Out[188]:
```
(16500, 1)
```

Verctorizing for Teacher Previously Posted Projected for Test Data -

In [189]:
```
import warnings
warnings.filterwarnings("ignore")

teacher_number_of_previously_posted_projects_scalar.fit(project_data_test['tea
cher_number_of_previously_posted_projects'].values.reshape(-1,1))
teacher_number_of_previously_posted_projects_scalar.transform(project_data_tes
t['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)) # find
ing the mean and standard deviation of this data
print(f"Mean : {teacher_number_of_previously_posted_projects_scalar.mean_[0]},
 Standard deviation : {np.sqrt(teacher_number_of_previously_posted_projects_sc
alar.var_[0])}")

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized_test = teacher_numbe
r_of_previously_posted_projects_scalar.transform(project_data_test['teacher_nu
mber_of_previously_posted_projects'].values.reshape(-1, 1))
```

```
Mean : 11.411515151515152, Standard deviation : 28.14762578084083
```

In [190]:
```
teacher_number_of_previously_posted_projects_standardized_test.shape
```

Out[190]:
```
(16500, 1)
```

In [191]:
```
project_data_train.columns
```

Out[191]:
```
Index(['index', 'id', 'teacher_id', 'project_submitted_datetime',
       'project_title', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'labels', 'Category_
0',
       'Category_1', 'level_0', 'SubCategory_0', 'SubCategory_1', 'essay',
       'State_Class_0', 'State_Class_1', 'Grade_0', 'Grade_1',
       'Prefix_Class_0', 'Prefix_Class_1', 'price', 'quantity'],
      dtype='object')
```

In [192]:
```
import warnings
warnings.filterwarnings("ignore")

Category_0 = StandardScaler()
Category_0.fit_transform(project_data_train['Category_0'].values.reshape(-1,1
)) # finding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
Category_0_train_standardized = Category_0.transform(project_data_train['Categ
ory_0'].values.reshape(-1, 1))
```

In [193]:
```
Category_1 = StandardScaler()
Category_1.fit_transform(project_data_train['Category_1'].values.reshape(-1,1
)) # finding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
Category_1_train_standardized = Category_1.transform(project_data_train['Categ
ory_0'].values.reshape(-1, 1))
```

In [194]:
```
SubCat_1 = StandardScaler()
SubCat_1.fit_transform(project_data_train['SubCategory_1'].values.reshape(-1,1
)) # finding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
SubCat_1_train_standardized = SubCat_1.transform(project_data_train['SubCatego
ry_1'].values.reshape(-1, 1))
```

In [195]:
```
SubCat_0 = StandardScaler()
SubCat_0.fit_transform(project_data_train['SubCategory_0'].values.reshape(-1,1
)) # finding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
SubCat_0_train_standardized = SubCat_0.transform(project_data_train['SubCatego
ry_0'].values.reshape(-1, 1))
```

In [196]:
```
State_0 = StandardScaler()
State_0.fit_transform(project_data_train['State_Class_0'].values.reshape(-1,1
)) # finding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
State_0_train_standardized = State_0.transform(project_data_train['State_Class
_0'].values.reshape(-1, 1))
```

In [197]:
```
State_1 = StandardScaler()
State_1.fit_transform(project_data_train['State_Class_1'].values.reshape(-1,1
)) # finding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
State_1_standardized = State_1.transform(project_data_train['State_Class_1'].v
alues.reshape(-1, 1))
```

In [198]:
```
Grade_1 = StandardScaler()
Grade_1.fit_transform(project_data_train['Grade_0'].values.reshape(-1,1)) # fi
nding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
Grade_1_standardized = Grade_1.transform(project_data_train['Grade_0'].values.
reshape(-1, 1))
```

In [199]:
```
Grade_0 = StandardScaler()
Grade_0.fit_transform(project_data_train['Grade_1'].values.reshape(-1,1)) # fi
nding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
Grade_0_standardized = Grade_0.transform(project_data_train['Grade_1'].values.
reshape(-1, 1))
```

In [201]:
```
Prefix_0 = StandardScaler()
Prefix_0.fit_transform(project_data_train['Prefix_Class_0'].values.reshape(-1,
1)) # finding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
Prefix_0_standardized = Prefix_0.transform(project_data_train['Prefix_Class_0'
].values.reshape(-1, 1))
```

In [202]:
```
Prefix_1 = StandardScaler()
Prefix_1.fit_transform(project_data_train['Prefix_Class_1'].values.reshape(-1,
1)) # finding the mean and standard deviation of this data
#print(f"Mean : {essays.mean_[0]}, Standard deviation : {np.sqrt(essays.var_
[0])}")

# Now standardize the data with above maen and variance.
Prefix_1_standardized = Prefix_1.transform(project_data_train['Prefix_Class_1'
].values.reshape(-1, 1))
```

Merging Features for BoW

In [229]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a den
se matirx :)
X_train = hstack((Category_1_train_standardized,Category_0_train_standardized,
SubCat_1_train_standardized,SubCat_0_train_standardized,State_0_train_standard
ized,Grade_1_standardized,Grade_0_standardized,State_1_standardized,Prefix_0_s
tandardized,Prefix_1_standardized,teacher_number_of_previously_posted_projects
_standardized,price_standardized,title_bow,text_bow)).tocsr()  #https://www.ka
ggle.com/c/quora-question-pairs/discussion/33491 taken from
X_train.shape
```

Out[229]: (33500, 20018)

In [205]:
```
Category_1_test_standardized = Category_1.transform(project_data_test['Categor
y_1'].values.reshape(-1, 1))
Category_0_test_standardized = Category_0.transform(project_data_test['Categor
y_0'].values.reshape(-1, 1))
SubCat_0_test_standardized = SubCat_0.transform(project_data_test['SubCategory
_0'].values.reshape(-1, 1))
SubCat_1_test_standardized = SubCat_1.transform(project_data_test['SubCategory
_1'].values.reshape(-1, 1))
State_0_test_standardized = State_0.transform(project_data_test['State_Class_
0'].values.reshape(-1, 1))
State_1_test_standardized = State_1.transform(project_data_test['State_Class_
1'].values.reshape(-1, 1))
Grade_0_test_standardized = Grade_0.transform(project_data_test['Grade_Class_
0'].values.reshape(-1, 1))
Grade_1_test_standardized = Grade_1.transform(project_data_test['Grade_Class_
1'].values.reshape(-1, 1))
Prefix_0_test_standardized = Prefix_0.transform(project_data_test['Prefix_Clas
s_0'].values.reshape(-1, 1))
Prefix_1_test_standardized = Prefix_1.transform(project_data_test['Prefix_Clas
s_1'].values.reshape(-1, 1))
```

In [230]:
```
X_test = hstack((Category_1_test_standardized,Category_0_test_standardized,Sub
Cat_0_test_standardized,SubCat_1_test_standardized,State_0_test_standardized,S
tate_1_test_standardized,Grade_0_test_standardized,Grade_1_test_standardized,P
refix_0_test_standardized,Prefix_1_test_standardized,teacher_number_of_previou
sly_posted_projects_standardized_test,price_standardized_test,text_bow_test,ti
tle_bow_test)).tocsr()  #https://www.kaggle.com/c/quora-question-pairs/discuss
ion/33491 taken from
X_test.shape
```

Out[230]: (16500, 20018)

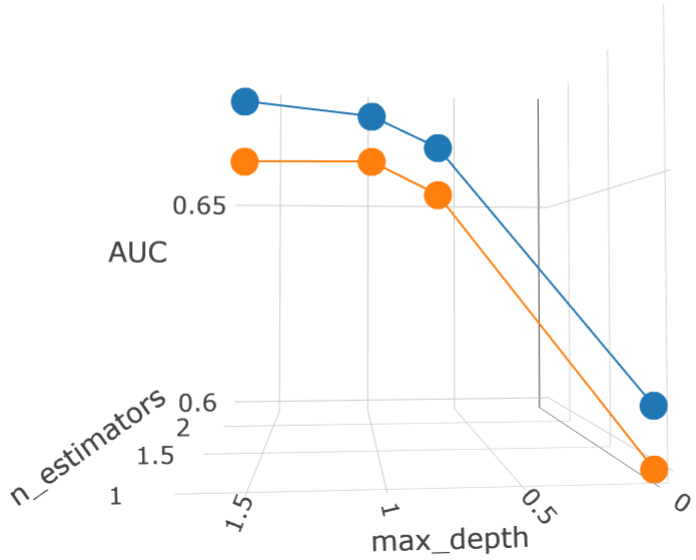
RF for BoW

In [210]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cross_validation import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.cross_validation import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import cross_validation
from math import log
```

In [211]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

C = RandomForestClassifier()

n_estimators=[10,50,100,200]
max_depth=[1, 5, 10, 50]

import math

log_max_depth = [math.log10(x) for x in max_depth]
log_n_estimators=[math.log10(x) for x in n_estimators]

print("Printing parameter Data and Corresponding Log value for Max Depth")
data={'Parameter value':max_depth,'Corresponding Log Value':log_max_depth}
param=pd.DataFrame(data)
print("="*100)
print(param)

print("Printing parameter Data and Corresponding Log value for Estimators")
data={'Parameter value':n_estimators,'Corresponding Log Value':log_n_estimators}
param=pd.DataFrame(data)
print("="*100)
print(param)

parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(C, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(X_train, labels_train)

#data={'Parameter value':[0.0001,0.001,0.01,0.1,1,5,10,20,30,40],'Corresponding Log Value':[log_my_data]}

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

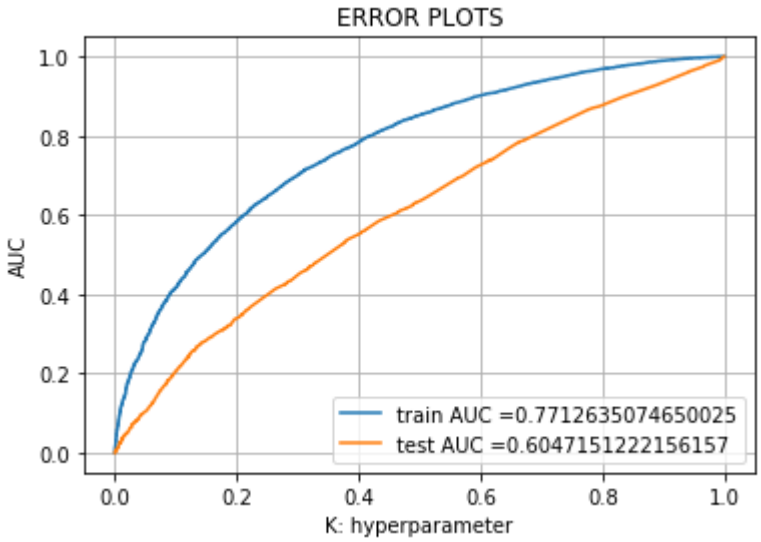
```
Printing parameter Data and Corresponding Log value for Max Depth
====================================================================================================
   Parameter value  Corresponding Log Value
0                1                  0.00000
1                5                  0.69897
2               10                  1.00000
3               50                  1.69897
Printing parameter Data and Corresponding Log value for Estimators
====================================================================================================
   Parameter value  Corresponding Log Value
0               10                  1.00000
1               50                  1.69897
2              100                  2.00000
3              200                  2.30103
```

In [212]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=log_n_estimators, y=log_max_depth, z=train_auc, name =
 'train')
trace2 = go.Scatter3d(x=log_n_estimators, y=log_max_depth, z=cv_auc, name = 'C
ross validation')
data = [trace1, trace2]

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

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



In [231]:
```
def model_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs

    y_data_pred = []
    y_data_pred.extend(clf.predict_proba(data[:])[:,1])

    return y_data_pred
```

In [255]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
#from sklearn.calibration import CalibratedClassifierCV


neigh = RandomForestClassifier(n_estimators=50,max_depth=5,class_weight='balanced')
neigh.fit(X_train, labels_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = model_predict(neigh, X_train)
y_test_pred = model_predict(neigh, X_test)


train_fpr, train_tpr, tr_thresholds = roc_curve(labels_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(labels_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```

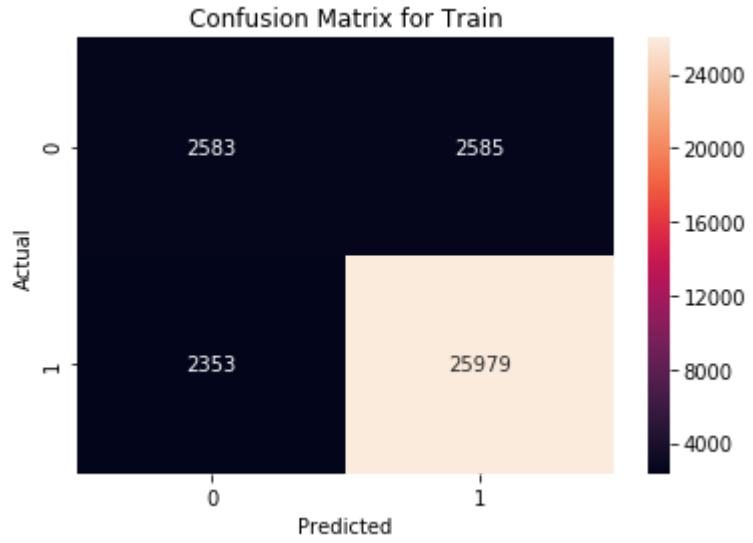


In [237]:
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [238]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm=confusion_matrix(labels_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr))
sns.heatmap(cm, annot=True, fmt="d" )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Train")
```

```
====================================================================================
======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999996255834908 for threshold 0.485
```

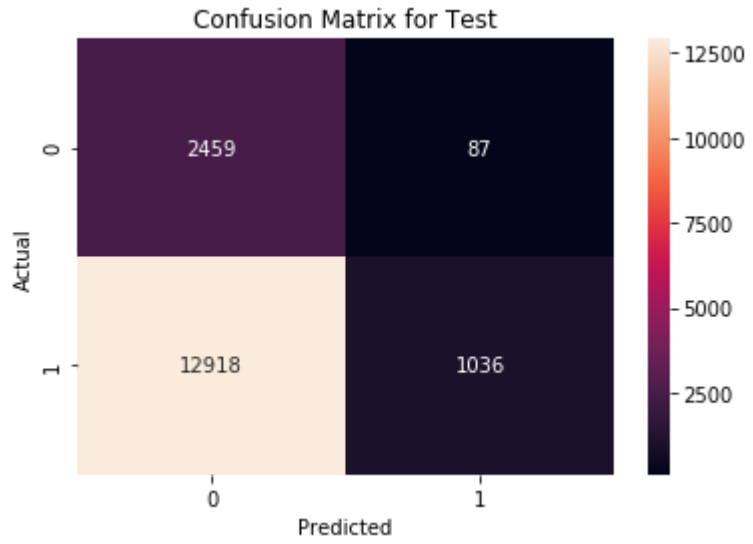
Out[238]: Text(0.5,1,'Confusion Matrix for Train')



In [239]:
```
print("Test confusion matrix")
cm1=confusion_matrix(labels_test, predict(y_test_pred, tr_thresholds, test_fpr
, test_fpr))
sns.heatmap(cm1, annot=True,fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Test")
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999984572938835 for threshold 0.497
```

Out[239]: Text(0.5,1,'Confusion Matrix for Test')

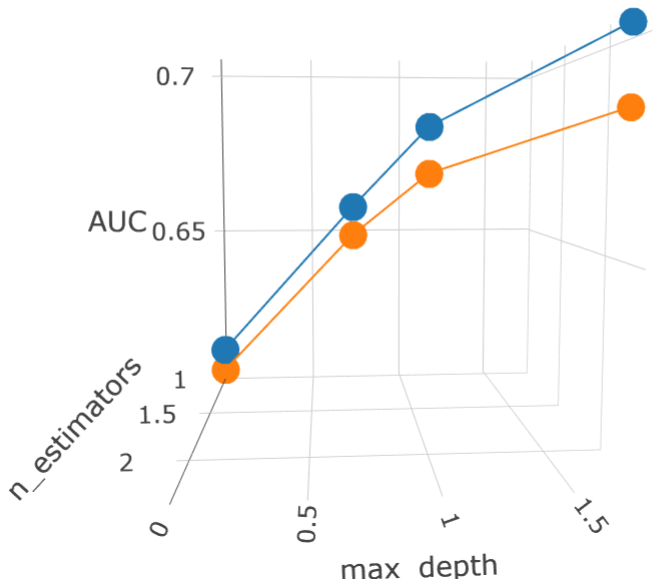


GBDT for BoW

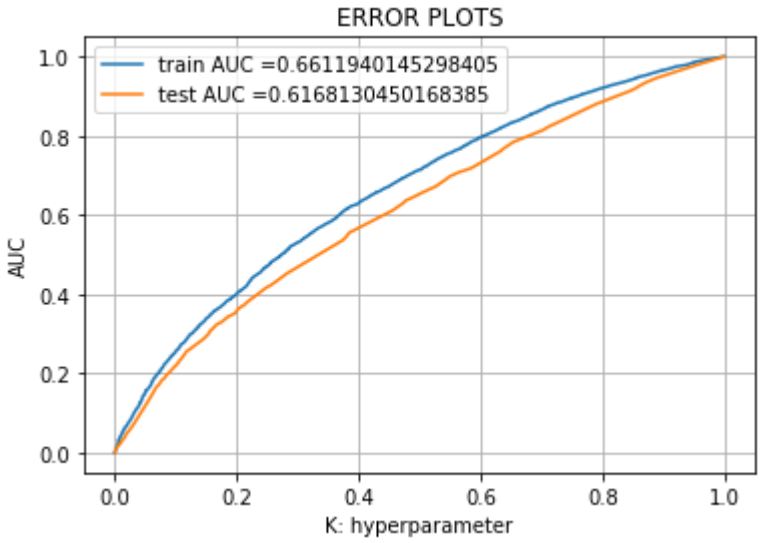
In [226]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier

C = GradientBoostingClassifier()

n_estimators=[10,50,100,200]
max_depth=[1, 5, 10, 50]

import math

log_max_depth = [math.log10(x) for x in max_depth]
log_n_estimators=[math.log10(x) for x in n_estimators]

print("Printing parameter Data and Corresponding Log value for Max Depth")
data={'Parameter value':max_depth,'Corresponding Log Value':log_max_depth}
param=pd.DataFrame(data)
print("="*100)
print(param)

print("Printing parameter Data and Corresponding Log value for Estimators")
data={'Parameter value':n_estimators,'Corresponding Log Value':log_n_estimators}
param=pd.DataFrame(data)
print("="*100)
print(param)

parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(C, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(X_train, labels_train)

#data={'Parameter value':[0.0001,0.001,0.01,0.1,1,5,10,20,30,40],'Corresponding Log Value':[log_my_data]}

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

```
Printing parameter Data and Corresponding Log value for Max Depth
====================================================================================================
   Parameter value  Corresponding Log Value
0                1                  0.00000
1                5                  0.69897
2               10                  1.00000
3               50                  1.69897
Printing parameter Data and Corresponding Log value for Estimators
====================================================================================================
   Parameter value  Corresponding Log Value
0               10                  1.00000
1               50                  1.69897
2              100                  2.00000
3              200                  2.30103
```

In [246]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=train_auc, name =
'train')
trace2 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=cv_auc, name = 'Cro
ss validation')
data = [trace1, trace2]

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

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



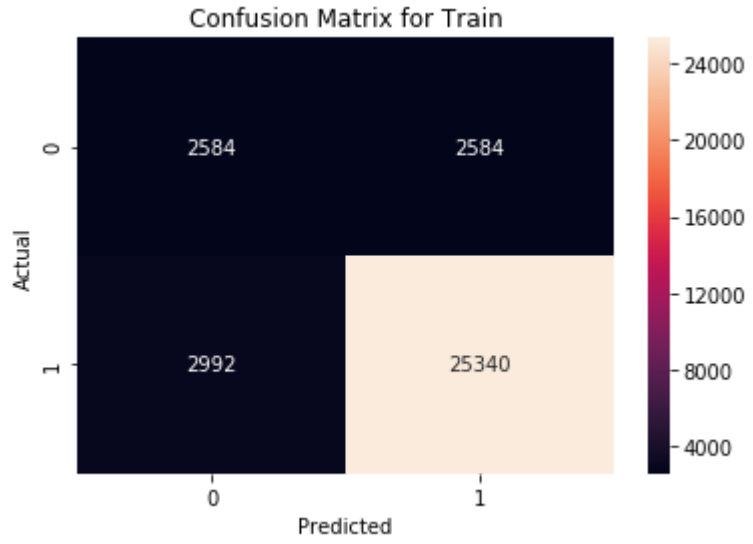
In [247]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.ensemble import GradientBoostingClassifier


neigh = GradientBoostingClassifier(n_estimators=50,max_depth=5)
neigh.fit(X_train, labels_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = model_predict(neigh, X_train)
y_test_pred = model_predict(neigh, X_test)


train_fpr, train_tpr, tr_thresholds = roc_curve(labels_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(labels_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [244]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm=confusion_matrix(labels_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr))
sns.heatmap(cm, annot=True, fmt="d" )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Train")
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.783
```

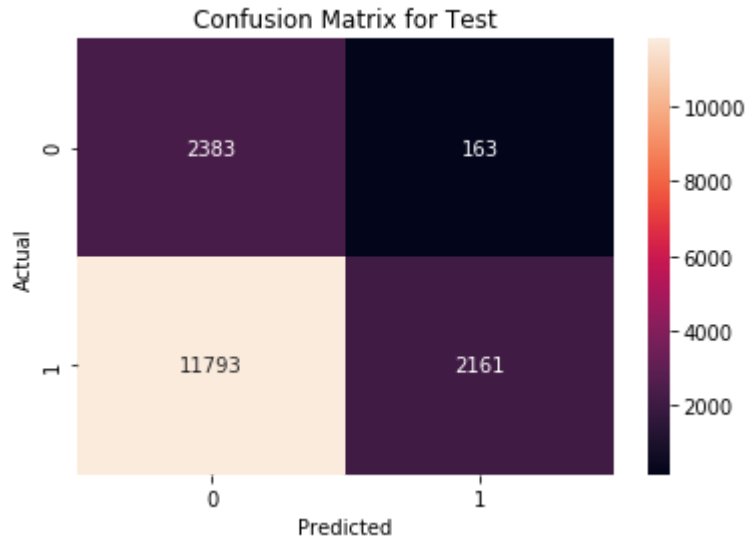
Out[244]: Text(0.5,1,'Confusion Matrix for Train')



In [245]:
```
print("Test confusion matrix")
cm1=confusion_matrix(labels_test, predict(y_test_pred, tr_thresholds, test_fpr
, test_fpr))
sns.heatmap(cm1, annot=True,fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Test")
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999938291755347 for threshold 0.87
```

Out[245]: Text(0.5,1,'Confusion Matrix for Test')



Merging Features for TFIDF

In [265]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a den
se matirx :)
X1_train = hstack((Category_1_train_standardized,Category_0_train_standardized
,SubCat_1_train_standardized,SubCat_0_train_standardized,State_0_train_standar
dized,Grade_1_standardized,Grade_0_standardized,State_1_standardized,Prefix_0_
standardized,Prefix_1_standardized,teacher_number_of_previously_posted_project
s_standardized,price_standardized,text_tfidf,tittle_tfidf)).tocsr()  #https://
www.kaggle.com/c/quora-question-pairs/discussion/33491 taken from
X1_train.shape
```

Out[265]: (33500, 10012)

In [266]:
```
X1_test = hstack((Category_1_test_standardized,Category_0_test_standardized,Su
bCat_0_test_standardized,SubCat_1_test_standardized,State_0_test_standardized,
State_1_test_standardized,Grade_0_test_standardized,Grade_1_test_standardized,
Prefix_0_test_standardized,Prefix_1_test_standardized,teacher_number_of_previo
usly_posted_projects_standardized_test,price_standardized_test,text_tfidf_test
,title_tfidf_test)).tocsr()  #https://www.kaggle.com/c/quora-question-pairs/di
scussion/33491 taken from
X1_test.shape
```

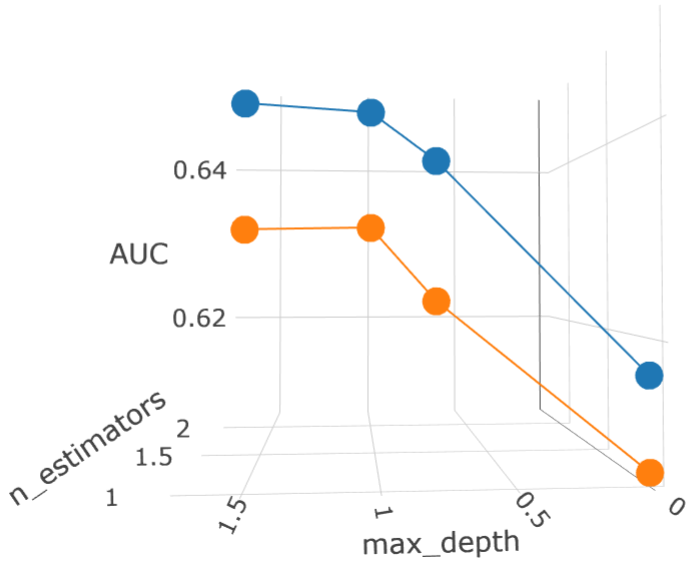
Out[266]: (16500, 10012)

RF for TFIDF

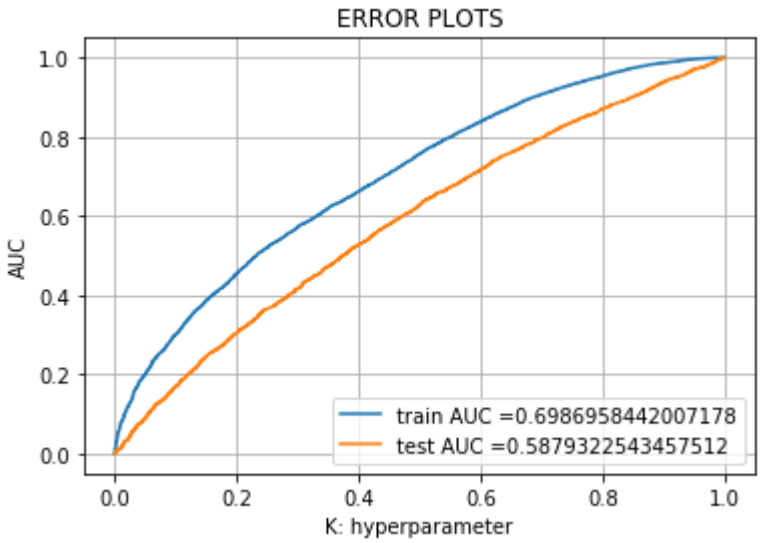
In [261]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

C = RandomForestClassifier()

n_estimators=[10,50,100,200]
max_depth=[1, 5, 10, 50]

import math

log_max_depth = [math.log10(x) for x in max_depth]
log_n_estimators=[math.log10(x) for x in n_estimators]

print("Printing parameter Data and Corresponding Log value for Max Depth")
data={'Parameter value':max_depth,'Corresponding Log Value':log_max_depth}
param=pd.DataFrame(data)
print("="*100)
print(param)

print("Printing parameter Data and Corresponding Log value for Estimators")
data={'Parameter value':n_estimators,'Corresponding Log Value':log_n_estimators}
param=pd.DataFrame(data)
print("="*100)
print(param)

parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(C, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(X1_train, labels_train)

#data={'Parameter value':[0.0001,0.001,0.01,0.1,1,5,10,20,30,40],'Corresponding Log Value':[log_my_data]}

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

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

# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

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

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

```
Printing parameter Data and Corresponding Log value for Max Depth
================================================================================
=======================
   Parameter value  Corresponding Log Value
0                1                 0.00000
1                5                 0.69897
2               10                 1.00000
3               50                 1.69897
Printing parameter Data and Corresponding Log value for Estimators
================================================================================
=======================
   Parameter value  Corresponding Log Value
0               10                 1.00000
1               50                 1.69897
2              100                 2.00000
3              200                 2.30103
```



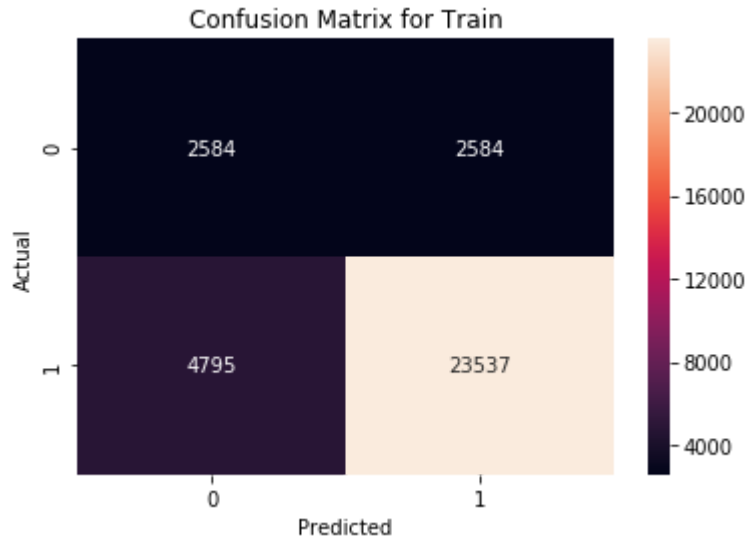
In [271]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
#from sklearn.calibration import CalibratedClassifierCV


neigh = RandomForestClassifier(n_estimators=50,max_depth=5,class_weight='balanced')
neigh.fit(X1_train, labels_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = model_predict(neigh, X1_train)
y_test_pred = model_predict(neigh, X1_test)


train_fpr, train_tpr, tr_thresholds = roc_curve(labels_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(labels_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [268]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm=confusion_matrix(labels_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr))
sns.heatmap(cm, annot=True, fmt="d" )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Train")
```

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

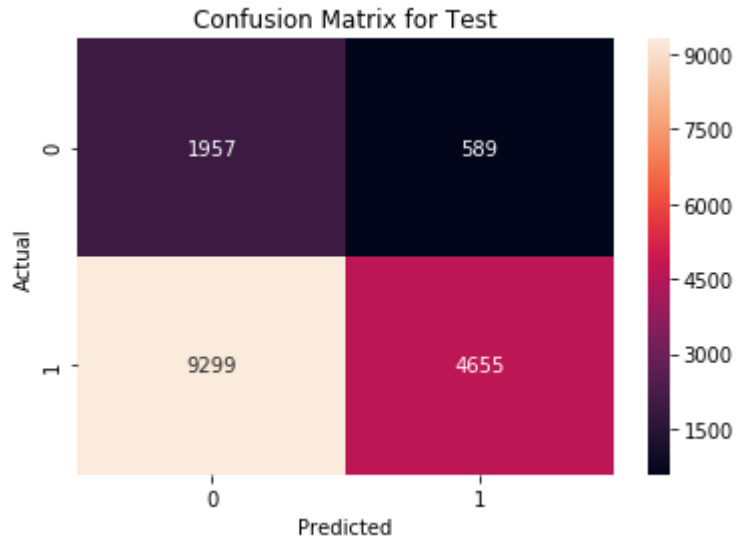
Out[268]: Text(0.5,1,'Confusion Matrix for Train')



In [269]:
```
print("Test confusion matrix")
cm1=confusion_matrix(labels_test, predict(y_test_pred, tr_thresholds, test_fpr
, test_fpr))
sns.heatmap(cm1, annot=True,fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Test")
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.514
```

Out[269]: Text(0.5,1,'Confusion Matrix for Test')

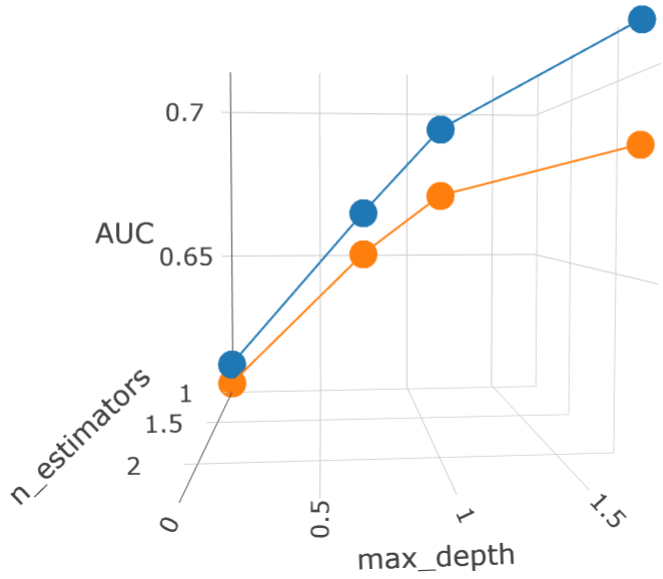


Gradient Boosting for TFIDF

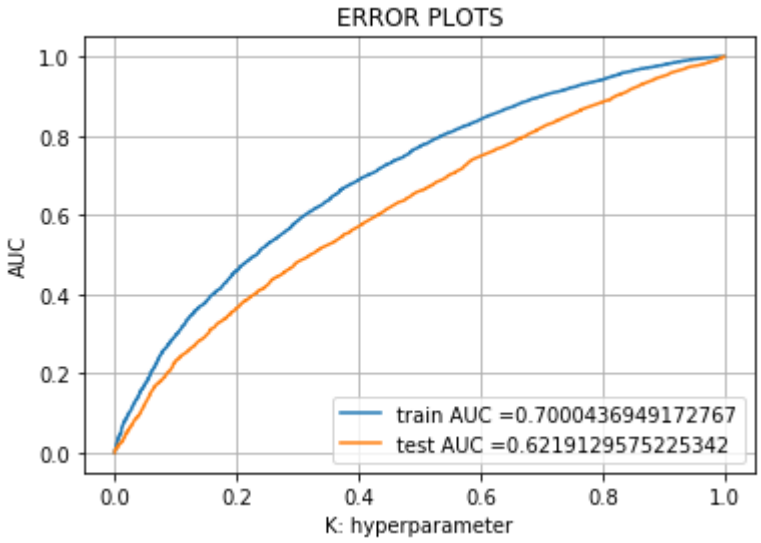
In [263]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier

C = GradientBoostingClassifier()

n_estimators=[10,50,100,200]
max_depth=[1, 5, 10, 50]

import math

log_max_depth = [math.log10(x) for x in max_depth]
log_n_estimators=[math.log10(x) for x in n_estimators]

print("Printing parameter Data and Corresponding Log value for Max Depth")
data={'Parameter value':max_depth,'Corresponding Log Value':log_max_depth}
param=pd.DataFrame(data)
print("="*100)
print(param)

print("Printing parameter Data and Corresponding Log value for Estimators")
data={'Parameter value':n_estimators,'Corresponding Log Value':log_n_estimators}
param=pd.DataFrame(data)
print("="*100)
print(param)

parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(C, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(X1_train, labels_train)

#data={'Parameter value':[0.0001,0.001,0.01,0.1,1,5,10,20,30,40],'Corresponding Log Value':[log_my_data]}

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

```
Printing parameter Data and Corresponding Log value for Max Depth
================================================================================
========================
   Parameter value  Corresponding Log Value
0                1                 0.00000
1                5                 0.69897
2               10                 1.00000
3               50                 1.69897
Printing parameter Data and Corresponding Log value for Estimators
================================================================================
========================
   Parameter value  Corresponding Log Value
0               10                 1.00000
1               50                 1.69897
2              100                 2.00000
3              200                 2.30103
```

In [264]:
```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=train_auc, name =
'train')
trace2 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=cv_auc, name = 'Cro
ss validation')
data = [trace1, trace2]

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

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



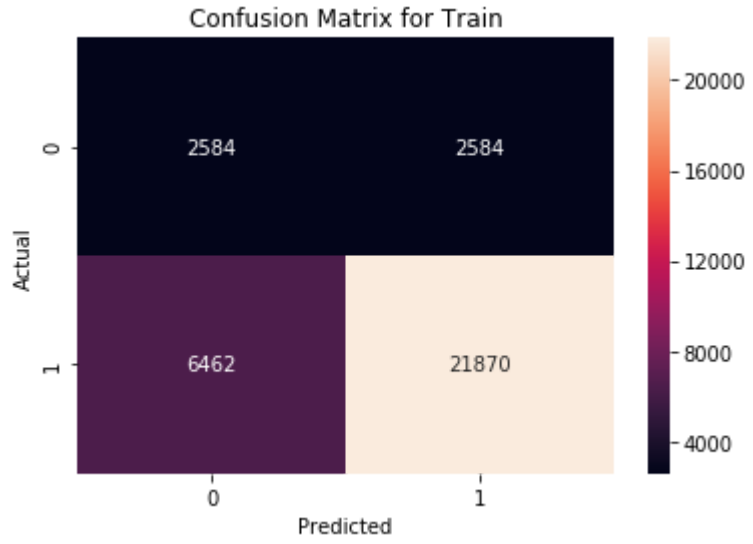
In [272]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
#from sklearn.calibration import CalibratedClassifierCV


neigh = GradientBoostingClassifier(n_estimators=50,max_depth=5)
neigh.fit(X1_train, labels_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = model_predict(neigh, X1_train)
y_test_pred = model_predict(neigh, X1_test)


train_fpr, train_tpr, tr_thresholds = roc_curve(labels_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(labels_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [273]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm=confusion_matrix(labels_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr))
sns.heatmap(cm, annot=True, fmt="d" )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Train")
```

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

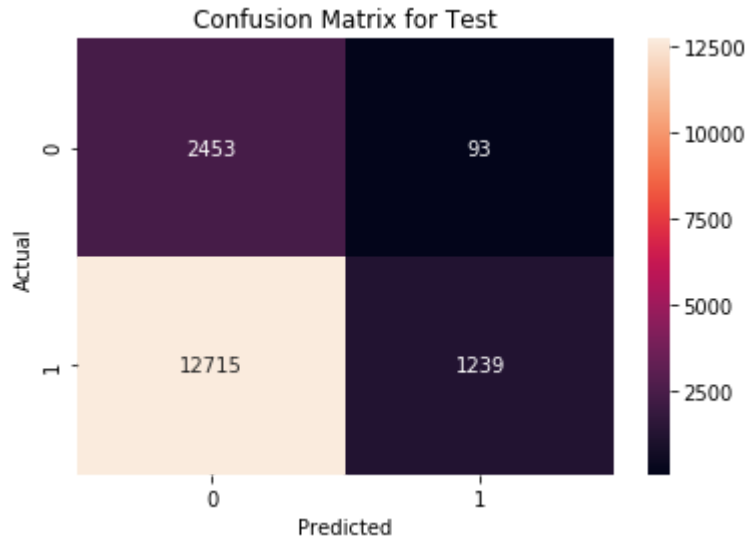
Out[273]: Text(0.5,1,'Confusion Matrix for Train')



In [274]:
```
print("Test confusion matrix")
cm1=confusion_matrix(labels_test, predict(y_test_pred, tr_thresholds, test_fpr
, test_fpr))
sns.heatmap(cm1, annot=True,fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Test")
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.89
```

Out[274]: Text(0.5,1,'Confusion Matrix for Test')



Merging Features for AVG W2V

In [283]:
```
from scipy.sparse import coo_matrix, hstack
X2_train = np.hstack(((Category_1_train_standardized),(Category_0_train_standa
rdized),(Grade_1_standardized),(Grade_0_standardized),(State_0_train_standardi
zed),(Prefix_0_standardized),(Prefix_1_standardized),(teacher_number_of_previo
usly_posted_projects_standardized),(price_standardized),(SubCat_1_train_standa
rdized),(SubCat_0_train_standardized),(State_1_standardized),avg_w2v_vectors,a
vg_w2v_vectors_tittle))
X2_train.shape
```

Out[283]: (33500, 612)

In [284]:
```
X2_test = np.hstack(((Category_1_test_standardized),(Category_0_test_standardi
zed),(SubCat_0_test_standardized),(SubCat_1_test_standardized),(State_0_test_s
tandardized),(State_1_test_standardized),(Grade_0_test_standardized),(Grade_1_
test_standardized),(Prefix_0_test_standardized),(Prefix_1_test_standardized),(
teacher_number_of_previously_posted_projects_standardized_test),(price_standar
dized_test),avg_w2v_vectors_test,avg_w2v_vectors_tittle_test))
X2_test.shape
```

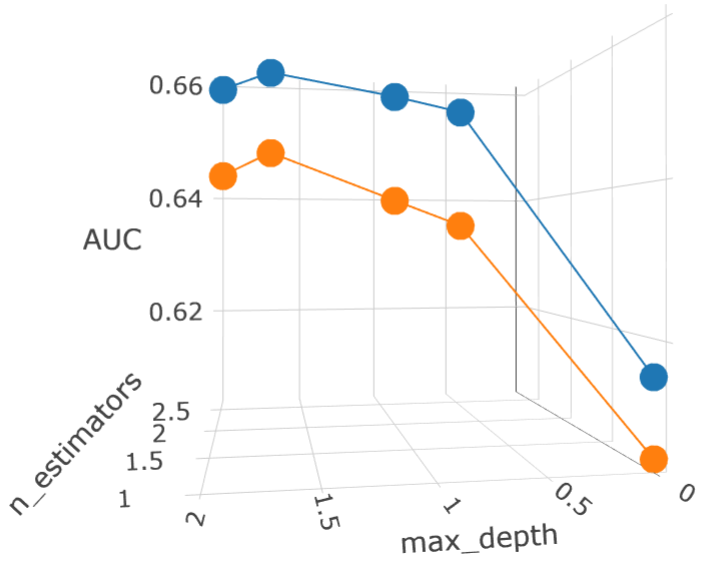
Out[284]: (16500, 612)

RF for AVG W2V

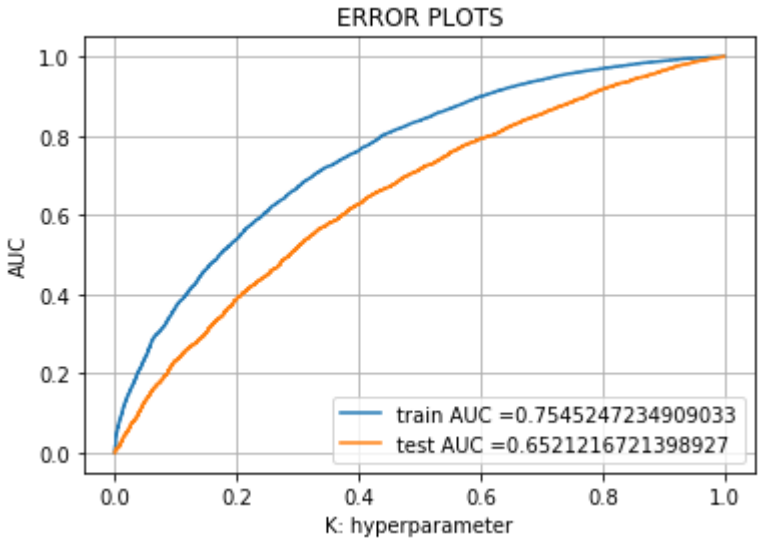
In [279]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

C = RandomForestClassifier()

n_estimators=[10,50,100,200,500]
max_depth=[1, 5, 10, 50, 100, 500, 100]

import math

log_max_depth = [math.log10(x) for x in max_depth]
log_n_estimators=[math.log10(x) for x in n_estimators]

print("Printing parameter Data and Corresponding Log value for Max Depth")
data={'Parameter value':max_depth,'Corresponding Log Value':log_max_depth}
param=pd.DataFrame(data)
print("="*100)
print(param)

print("Printing parameter Data and Corresponding Log value for Estimators")
data={'Parameter value':n_estimators,'Corresponding Log Value':log_n_estimators}
param=pd.DataFrame(data)
print("="*100)
print(param)

parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(C, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(X2_train, labels_train)

#data={'Parameter value':[0.0001,0.001,0.01,0.1,1,5,10,20,30,40],'Corresponding Log Value':[log_my_data]}

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

```
Printing parameter Data and Corresponding Log value for Max Depth
================================================================================
========================
   Parameter value  Corresponding Log Value
0                1                 0.00000
1                5                 0.69897
2               10                 1.00000
3               50                 1.69897
4              100                 2.00000
5              500                 2.69897
6              100                 2.00000
Printing parameter Data and Corresponding Log value for Estimators
================================================================================
========================
   Parameter value  Corresponding Log Value
0               10                 1.00000
1               50                 1.69897
2              100                 2.00000
3              200                 2.30103
4              500                 2.69897
```

In [280]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

trace1 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=train_auc, name =
'train')
trace2 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=cv_auc, name = 'Cro
ss validation')
data = [trace1, trace2]

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

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



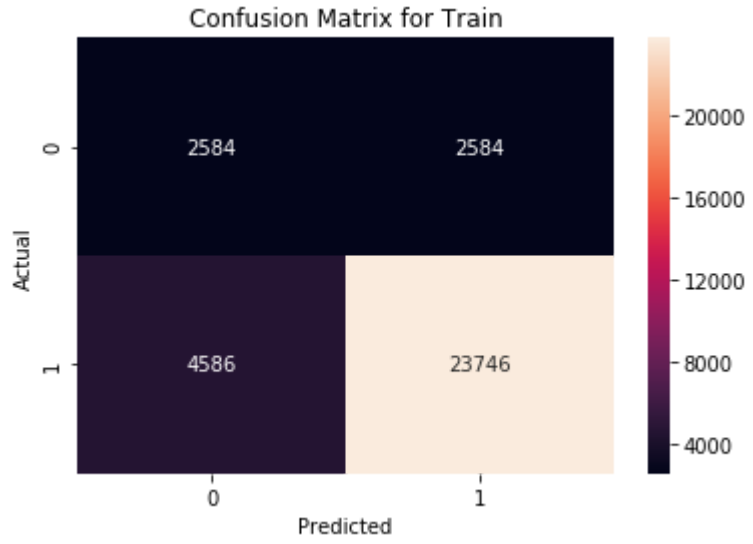
In [285]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
#from sklearn.calibration import CalibratedClassifierCV


neigh = RandomForestClassifier(n_estimators=50,max_depth=5,class_weight='balanced')
neigh.fit(X2_train, labels_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = model_predict(neigh, X2_train)
y_test_pred = model_predict(neigh, X2_test)


train_fpr, train_tpr, tr_thresholds = roc_curve(labels_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(labels_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [286]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm=confusion_matrix(labels_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr))
sns.heatmap(cm, annot=True, fmt="d" )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Train")
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.479
```

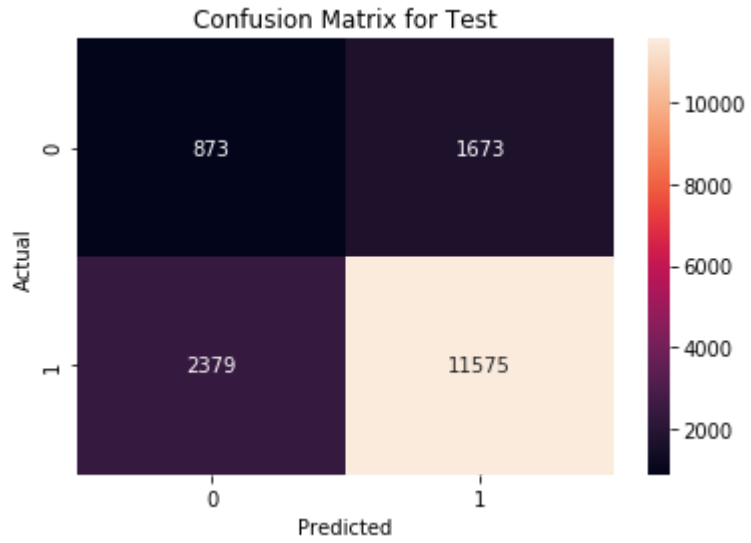
Out[286]: Text(0.5,1,'Confusion Matrix for Train')



In [287]:
```
print("Test confusion matrix")
cm1=confusion_matrix(labels_test, predict(y_test_pred, tr_thresholds, test_fpr
, test_fpr))
sns.heatmap(cm1, annot=True,fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Test")
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.522
```

Out[287]: Text(0.5,1,'Confusion Matrix for Test')

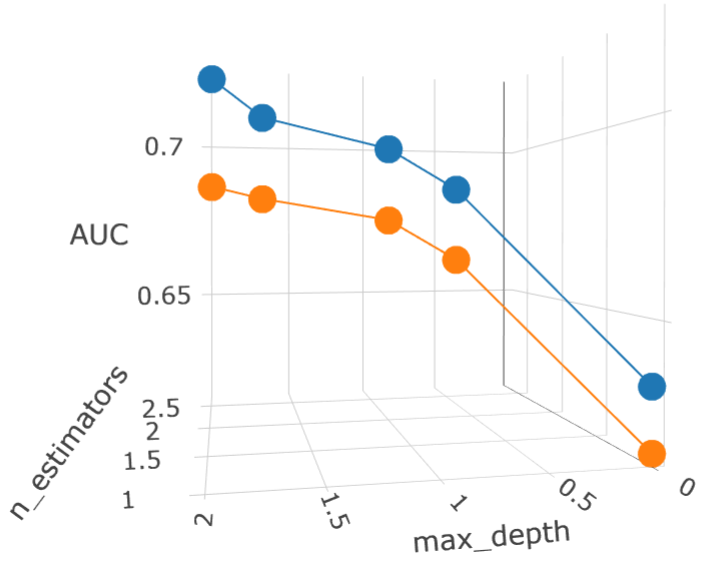


Gradient Boosting Classifier

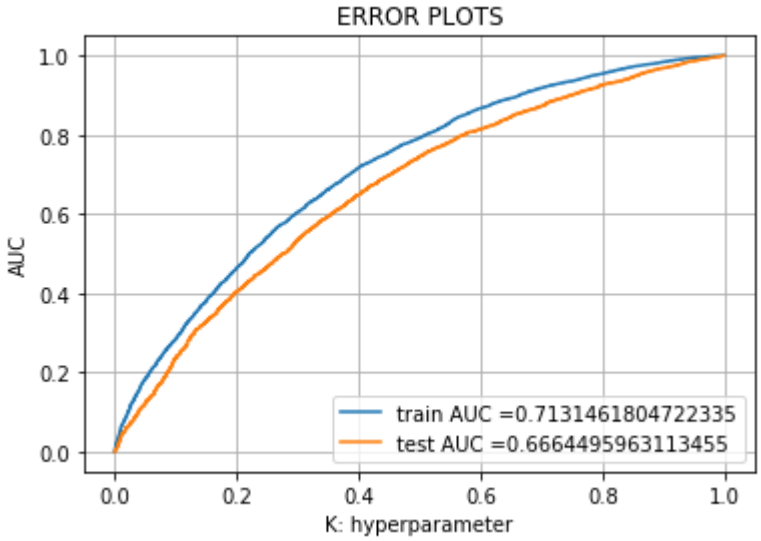
In [281]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#from sklearn.model_sel


#ection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier

C = GradientBoostingClassifier()

n_estimators=[10,50,100,200,500]
max_depth=[1, 5, 10, 50, 100, 500]

import math

log_max_depth = [math.log10(x) for x in max_depth]
log_n_estimators=[math.log10(x) for x in n_estimators]

print("Printing parameter Data and Corresponding Log value for Max Depth")
data={'Parameter value':max_depth,'Corresponding Log Value':log_max_depth}
param=pd.DataFrame(data)
print("="*100)
print(param)

print("Printing parameter Data and Corresponding Log value for Estimators")
data={'Parameter value':n_estimators,'Corresponding Log Value':log_n_estimators}
param=pd.DataFrame(data)
print("="*100)
print(param)

parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(C, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(X2_train, labels_train)

#data={'Parameter value':[0.0001,0.001,0.01,0.1,1,5,10,20,30,40],'Corresponding Log Value':[log_my_data]}

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

```
Printing parameter Data and Corresponding Log value for Max Depth
================================================================================
========================
   Parameter value  Corresponding Log Value
0                1                 0.00000
1                5                 0.69897
2               10                 1.00000
3               50                 1.69897
4              100                 2.00000
5              500                 2.69897
Printing parameter Data and Corresponding Log value for Estimators
================================================================================
========================
   Parameter value  Corresponding Log Value
0               10                 1.00000
1               50                 1.69897
2              100                 2.00000
3              200                 2.30103
4              500                 2.69897
```

In [282]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

trace1 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=train_auc, name =
'train')
trace2 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=cv_auc, name = 'Cro
ss validation')
data = [trace1, trace2]

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

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



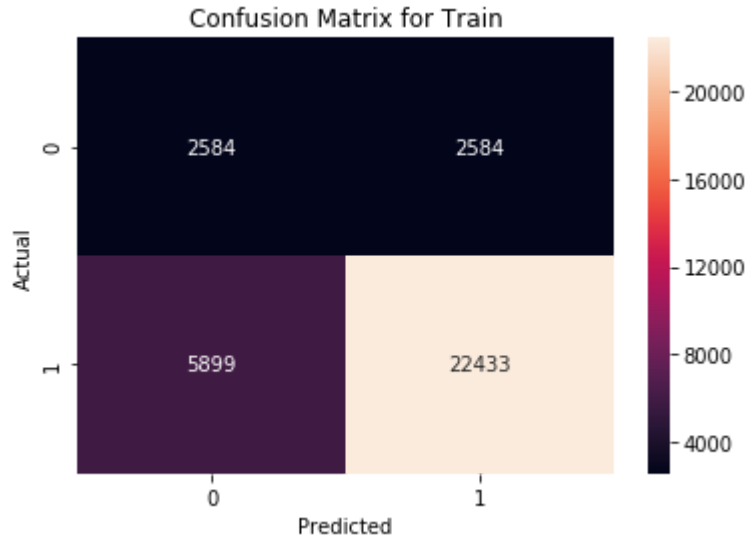
In [288]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
#from sklearn.calibration import CalibratedClassifierCV


neigh = GradientBoostingClassifier(n_estimators=50,max_depth=5)
neigh.fit(X2_train, labels_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = model_predict(neigh, X2_train)
y_test_pred = model_predict(neigh, X2_test)


train_fpr, train_tpr, tr_thresholds = roc_curve(labels_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(labels_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [289]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm=confusion_matrix(labels_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr))
sns.heatmap(cm, annot=True, fmt="d" )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Train")
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.819
```

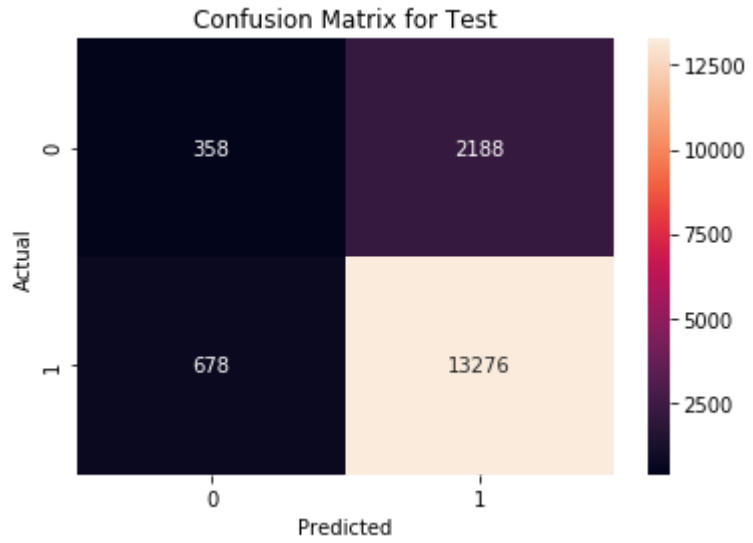
Out[289]: Text(0.5,1,'Confusion Matrix for Train')



In [290]:
```
print("Test confusion matrix")
cm1=confusion_matrix(labels_test, predict(y_test_pred, tr_thresholds, test_fpr
, test_fpr))
sns.heatmap(cm1, annot=True,fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Test")
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.86
```

Out[290]: Text(0.5,1,'Confusion Matrix for Test')



RF for TFIDF W2V

In [305]:
```
from scipy.sparse import coo_matrix, hstack
X3_train = np.hstack(((Category_1_train_standardized),(Category_0_train_standa
rdized),(Grade_1_standardized),(Grade_0_standardized),(State_0_train_standardi
zed),(Prefix_0_standardized),(Prefix_1_standardized),(teacher_number_of_previo
usly_posted_projects_standardized),(price_standardized),(SubCat_1_train_standa
rdized),(SubCat_0_train_standardized),(State_1_standardized),tfidf_w2v_vectors
,tfidf_w2v_vectors_Title))
X3_train.shape
```

Out[305]: (33500, 612)

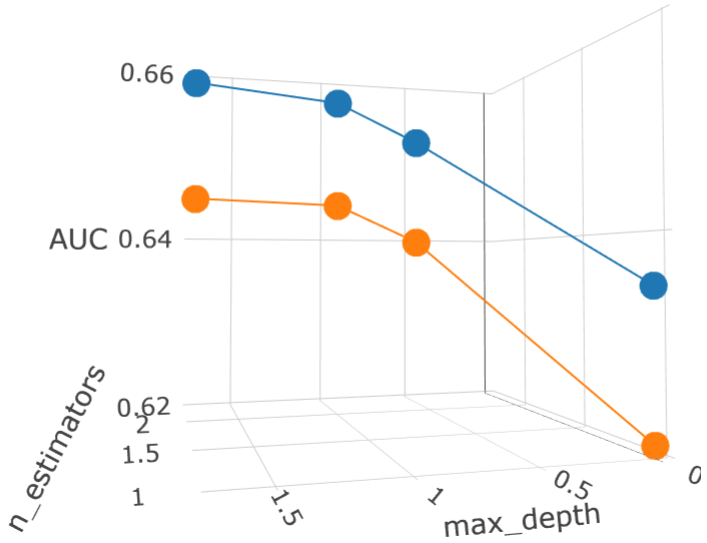
In [306]:
```
X3_test = np.hstack(((Category_1_test_standardized),(Category_0_test_standardi
zed),(SubCat_0_test_standardized),(SubCat_1_test_standardized),(State_0_test_s
tandardized),(State_1_test_standardized),(Grade_0_test_standardized),(Grade_1_
test_standardized),(Prefix_0_test_standardized),(Prefix_1_test_standardized),(
teacher_number_of_previously_posted_projects_standardized_test),(price_standar
dized_test),tfidf_w2v_vectors_test,tfidf_w2v_vectors_Title_test))
X3_test.shape
```

Out[306]: (16500, 612)

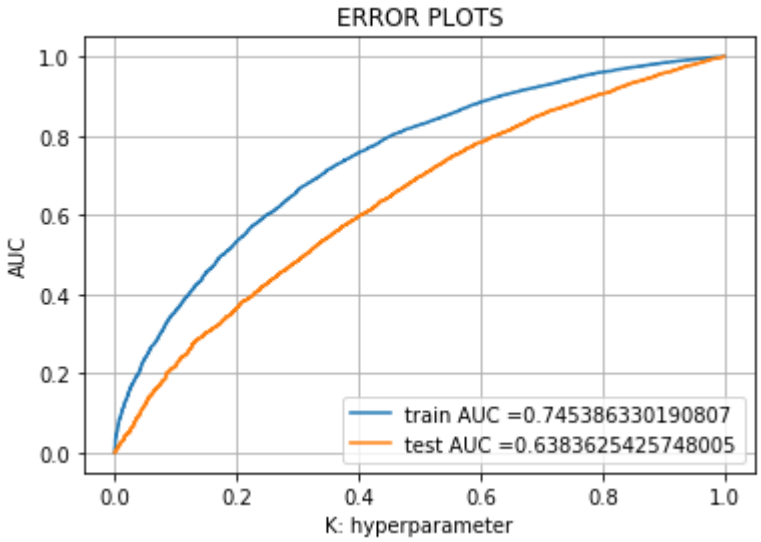
In [298]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

C = RandomForestClassifier()


n_estimators=[10,50,100,200]
max_depth=[1, 5, 10, 50]

import math

log_max_depth = [math.log10(x) for x in max_depth]
log_n_estimators=[math.log10(x) for x in n_estimators]

print("Printing parameter Data and Corresponding Log value for Max Depth")
data={'Parameter value':max_depth,'Corresponding Log Value':log_max_depth}
param=pd.DataFrame(data)
print("="*100)
print(param)

print("Printing parameter Data and Corresponding Log value for Estimators")
data={'Parameter value':n_estimators,'Corresponding Log Value':log_n_estimators}
param=pd.DataFrame(data)
print("="*100)
print(param)

parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(C, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(X3_train, labels_train)

#data={'Parameter value':[0.0001,0.001,0.01,0.1,1,5,10,20,30,40],'Corresponding Log Value':[log_my_data]}

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

```
Printing parameter Data and Corresponding Log value for Max Depth
====================================================================================================
   Parameter value  Corresponding Log Value
0                1                  0.00000
1                5                  0.69897
2               10                  1.00000
3               50                  1.69897
Printing parameter Data and Corresponding Log value for Estimators
====================================================================================================
   Parameter value  Corresponding Log Value
0               10                  1.00000
1               50                  1.69897
2              100                  2.00000
3              200                  2.30103
```

In [299]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

trace1 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=train_auc, name =
'train')
trace2 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=cv_auc, name = 'Cro
ss validation')
data = [trace1, trace2]

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

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



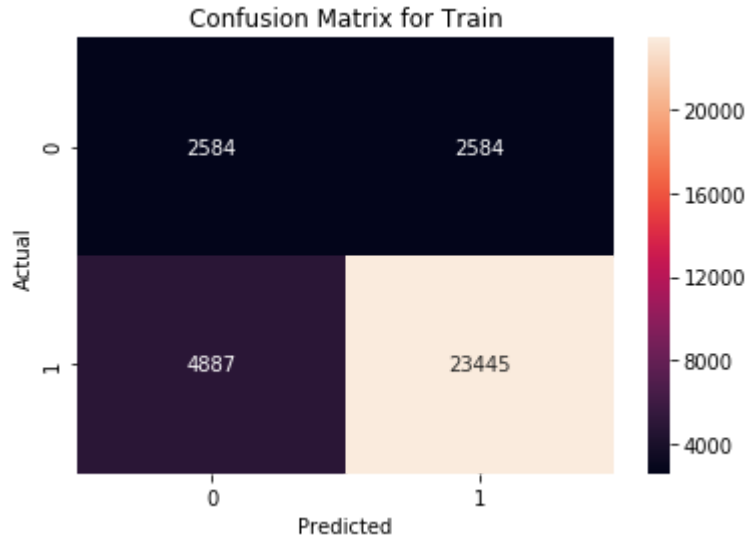
In [307]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
#from sklearn.calibration import CalibratedClassifierCV


neigh = RandomForestClassifier(n_estimators=50,max_depth=5,class_weight='balanced')
neigh.fit(X3_train, labels_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = model_predict(neigh, X3_train)
y_test_pred = model_predict(neigh, X3_test)


train_fpr, train_tpr, tr_thresholds = roc_curve(labels_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(labels_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [308]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm=confusion_matrix(labels_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr))
sns.heatmap(cm, annot=True, fmt="d" )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Train")
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.471
```

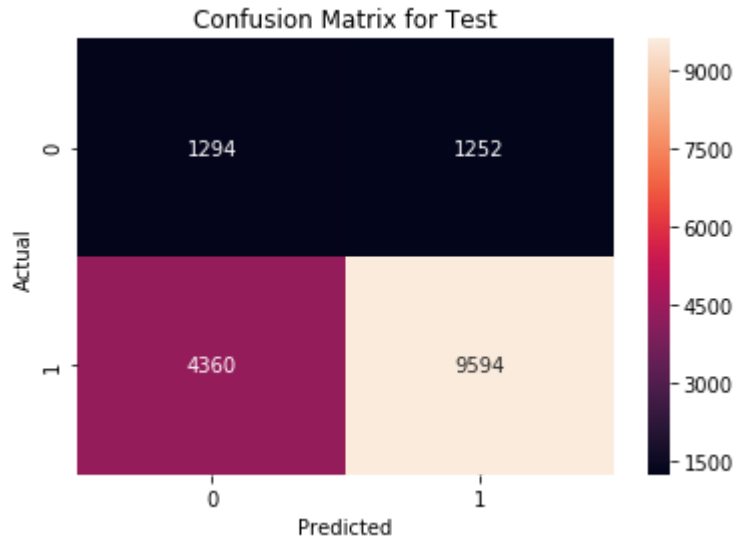
Out[308]: Text(0.5,1,'Confusion Matrix for Train')



In [309]:
```
print("Test confusion matrix")
cm1=confusion_matrix(labels_test, predict(y_test_pred, tr_thresholds, test_fpr
, test_fpr))
sns.heatmap(cm1, annot=True,fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Test")
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.518
```

Out[309]: Text(0.5,1,'Confusion Matrix for Test')

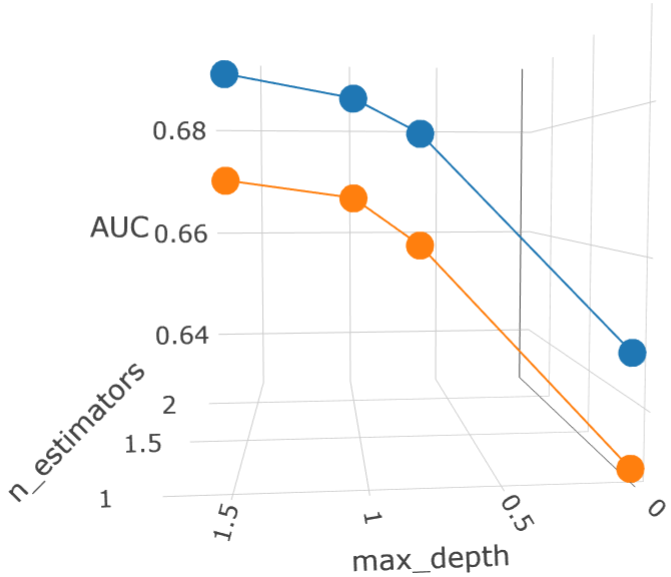


Gradient Boosting

In [300]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
#from sklearn.model_sel


#ection import GridSearchCV
from sklearn.ensemble import GradientBoostingClassifier

C = GradientBoostingClassifier()


n_estimators=[10,50,100,200]
max_depth=[1, 5, 10, 50]

import math

log_max_depth = [math.log10(x) for x in max_depth]
log_n_estimators=[math.log10(x) for x in n_estimators]

print("Printing parameter Data and Corresponding Log value for Max Depth")
data={'Parameter value':max_depth,'Corresponding Log Value':log_max_depth}
param=pd.DataFrame(data)
print("="*100)
print(param)

print("Printing parameter Data and Corresponding Log value for Estimators")
data={'Parameter value':n_estimators,'Corresponding Log Value':log_n_estimators}
param=pd.DataFrame(data)
print("="*100)
print(param)

parameters = {'n_estimators':n_estimators, 'max_depth':max_depth}
clf = GridSearchCV(C, parameters, cv=3, scoring='roc_auc',n_jobs=-1)
clf.fit(X3_train, labels_train)

#data={'Parameter value':[0.0001,0.001,0.01,0.1,1,5,10,20,30,40],'Corresponding Log Value':[log_my_data]}

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

```
Printing parameter Data and Corresponding Log value for Max Depth
================================================================================
========================
   Parameter value  Corresponding Log Value
0                1                 0.00000
1                5                 0.69897
2               10                 1.00000
3               50                 1.69897
Printing parameter Data and Corresponding Log value for Estimators
================================================================================
========================
   Parameter value  Corresponding Log Value
0               10                 1.00000
1               50                 1.69897
2              100                 2.00000
3              200                 2.30103
```

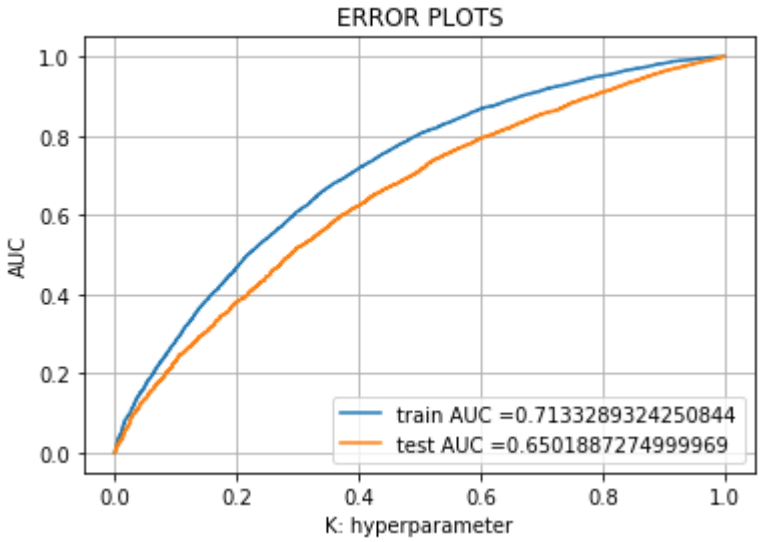
In [301]:
```
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
import numpy as np

trace1 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=train_auc, name =
'train')
trace2 = go.Scatter3d(x=log_n_estimators,y=log_max_depth,z=cv_auc, name = 'Cro
ss validation')
data = [trace1, trace2]

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

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



In [304]:
```
print(len(train_auc))
```

```
16
```

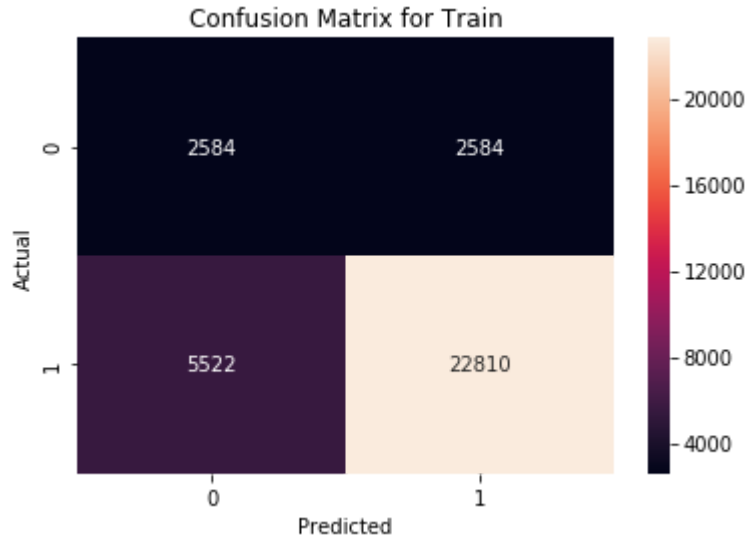
In [310]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
#from sklearn.calibration import CalibratedClassifierCV


neigh = GradientBoostingClassifier(n_estimators=50,max_depth=2)
neigh.fit(X3_train, labels_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs

y_train_pred = model_predict(neigh, X3_train)
y_test_pred = model_predict(neigh, X3_test)


train_fpr, train_tpr, tr_thresholds = roc_curve(labels_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(labels_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [311]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm=confusion_matrix(labels_train, predict(y_train_pred, tr_thresholds, train_f
pr, train_fpr))
sns.heatmap(cm, annot=True, fmt="d" )
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Train")
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.814
```

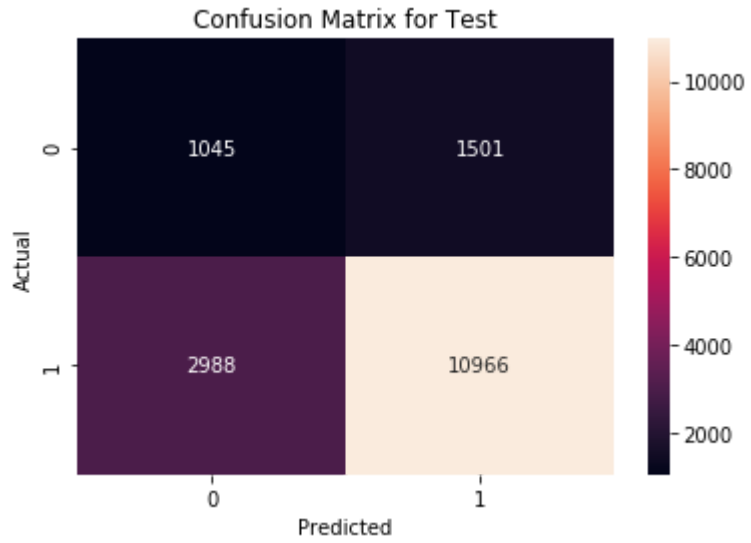
Out[311]: Text(0.5,1,'Confusion Matrix for Train')



In [312]:
```
print("Test confusion matrix")
cm1=confusion_matrix(labels_test, predict(y_test_pred, tr_thresholds, test_fpr
, test_fpr))
sns.heatmap(cm1, annot=True,fmt="d")
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title("Confusion Matrix for Test")
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.861
```

Out[312]: Text(0.5,1,'Confusion Matrix for Test')



In [313]:
```
from prettytable import PrettyTable


print("Pretty Table for Random Forest")
print("--"*50)

x = PrettyTable()

x.field_names = ["Vectorizer", "Model","n_estimators", "Max_depth", "Train AU
C", "Test AUC"]

x.add_row(["BOW", "Brute(Decision Tree)","100", 5 , 75, 59])
x.add_row(["TFIDF", "Brute(Decision Tree)","100", 7, 74, 60])
x.add_row(["AVG W2V", "Brute(Decision Tree)","50", 5, 75, 64 ])
x.add_row(["TFIDF W2V", "Brute(Decision Tree)","50", 5, 68, 63 ])
print(x)
```

```
Pretty Table for Random Forest
------------------------------------------------------------------------------
----------------------
+------------+----------------------+-------------+-----------+-----------+-
---------+
| Vectorizer |        Model         | n_estimators | Max_depth | Train AUC |
Test AUC |
+------------+----------------------+-------------+-----------+-----------+-
---------+
|    BOW     | Brute(Decision Tree) |     100      |     5     |     75    |
   59    |
|   TFIDF    | Brute(Decision Tree) |     100      |     7     |     74    |
   60    |
|  AVG W2V   | Brute(Decision Tree) |      50      |     5     |     75    |
   64    |
| TFIDF W2V  | Brute(Decision Tree) |      50      |     5     |     68    |
   63    |
+------------+----------------------+-------------+-----------+-----------+-
---------+
```

In [314]:
```
from prettytable import PrettyTable


print("Pretty Table for GBDT")
print("--"*50)

x = PrettyTable()

x.field_names = ["Vectorizer", "Model","n_estimators", "Max_depth", "Train AU
C", "Test AUC"]

x.add_row(["BOW", "Brute(Decision Tree)","50", 1 , 66, 61])
x.add_row(["TFIDF", "Brute(Decision Tree)","50", 2, 70, 62])
x.add_row(["AVG W2V", "Brute(Decision Tree)","50", 2, 71, 61 ])
x.add_row(["TFIDF W2V", "Brute(Decision Tree)","50", 2, 72, 63 ])
print(x)
```

```
Pretty Table for GBDT
----------------------------------------------------------------------------
----------------------
+------------+----------------------+--------------+-----------+-----------+-
---------+
| Vectorizer |        Model         | n_estimators | Max_depth | Train AUC |
Test AUC |
+------------+----------------------+--------------+-----------+-----------+-
---------+
|    BOW     | Brute(Decision Tree) |      50      |     1     |     66    |
   61    |
|   TFIDF    | Brute(Decision Tree) |      50      |     2     |     70    |
   62    |
|  AVG W2V   | Brute(Decision Tree) |      50      |     2     |     71    |
   61    |
| TFIDF W2V  | Brute(Decision Tree) |      50      |     2     |     72    |
   63    |
+------------+----------------------+--------------+-----------+-----------+-
---------+
```