Introduction

Donorschoose.org is a US-based non-profit organization that allows individuals to donate directly to public school classroom projects.Founded in 2000 by former public school teacher Charles Best, DonorsChoose.org was among the first civic crowdfunding platforms of its kind. The organization has been given Charity Navigator's highest rating every year since 2005.In January 2018, they announced that 1 million projects had been funded.To get students what they need to learn, the team at DonorsChoose.org needs to be able to connect donors with the projects that most inspire them.

Problem Statement

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 assignment 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.

Importing Libraries

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
```

```
offline.init_notebook_mode()
from collections import Counter
```

```
C:\Users\aksha\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning: paramiko missing, op
ening SSH/SCP/SFTP paths will be disabled.  `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip install
paramiko` to suppress')
C:\Users\aksha\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; al
iasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

Directory List

In [2]:

```
import os
os.chdir("D:\\applied AI\\Donorchoose")
```

About the dataset

The train_data.csv is the dataset provided by the DonorsChoose containin features as follows :-

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*

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

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

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

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

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

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

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

Notes on the Essay Data

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

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_4:__ "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.

Reading the data

In [3]:

```
train_data=pd.read_csv("train_data.csv")
res_data=pd.read_csv("resources.csv")
```

In [4]:

```
print("number of datapoints=",train_data.shape) #shape will tell us the number of projects we have
which is 109248
```

```
print("columns/atrributes name=",train_data.columns)
print(train_data.head(3))
```

```
number of datapoints= (109248, 17)
columns/atrributes name= Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix',
'school_state',
       'project_submitted_datetime', 'project_grade_category',
       'project_subject_categories', 'project_subject_subcategories',
       'project_title', 'project_essay_1', 'project_essay_2',
       'project_essay_3', 'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved'],
      dtype='object')
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.
2       21895  p182444  3465aaf82da834c0582ebd0ef8040ca0            Ms.

  school_state project_submitted_datetime project_grade_category  \
0           IN        2016-12-05 13:43:57          Grades PreK-2
1           FL        2016-10-25 09:22:10             Grades 6-8
2           AZ        2016-08-31 12:03:56             Grades 6-8

          project_subject_categories     project_subject_subcategories  \
0                Literacy & Language                     ESL, Literacy
1  History & Civics, Health & Sports  Civics & Government, Team Sports
2                    Health & Sports    Health & Wellness, Team Sports

                                       project_title  \
0   Educational Support for English Learners at Home
1              Wanted: Projector for Hungry Learners
2  Soccer Equipment for AWESOME Middle School Stu...

                                     project_essay_1  \
0  My students are English learners that are work...
1  Our students arrive to our school eager to lea...
2  \r\n\"True champions aren't always the ones th...

                                     project_essay_2 project_essay_3  \
0  \"The limits of your language are the limits o...             NaN
1  The projector we need for our school is very c...             NaN
2  The students on the campus come to school know...             NaN

  project_essay_4                           project_resource_summary  \
0             NaN  My students need opportunities to practice beg...
1             NaN  My students need a projector to help with view...
2             NaN  My students need shine guards, athletic socks,...

   teacher_number_of_previously_posted_projects  project_is_approved
0                                             0                    0
1                                             7                    1
2                                             1                    0
```

In [5]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
# Replacing datetime columns to date column
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(train_data.columns)] #if x e
ncounters column name project_submitted_datetime it will replace by date
#so a new column Date is created

#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40-84039
train_data['Date'] = pd.to_datetime(train_data['project_submitted_datetime']) #pd.to_datetime
converts argument to datetime
train_data.drop('project_submitted_datetime', axis=1, inplace=True) #dropping the column
project_submitted_date
train_data.sort_values(by=['Date'], inplace=True)#sorting the dataframe by date


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
train_data = train_data[cols] #adding the new column


train_data.head(2) #displaying the dataframe
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016-04-27 00:27:36	Grades PreK-2	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5	

In [6]:

```
print("datapoints in resources=",res_data.shape)
print("attributes of resources=",res_data.columns)
print(res_data.head(3))
```

```
datapoints in resources= (1541272, 4)
attributes of resources= Index(['id', 'description', 'quantity', 'price'], dtype='object')
        id                                        description  quantity  \
0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
1  p069063        Bouncy Bands for Desks (Blue support pipes)         3
2  p069063  Cory Stories: A Kid's Book About Living With Adhd         1

    price
0  149.00
1   14.95
2    8.45
```

In [7]:

```
#Refer-> https://www.shanelynn.ie/summarising-aggregation-and-grouping-data-in-python-pandas/

price_data = res_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index() #grouping
is done on the basis of ids and aggreating the sum of price and quantity column

#https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.merge.html?
highlight=merge#pandas.merge
train_data = train_data.merge(price_data, on='id', how='left')
print(train_data.head(1))
```

```
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0        8393  p205479  2bf07ba08945e5d8b2a3f269b2b3cfe5           Mrs.

  school_state                 Date project_grade_category  \
0           CA 2016-04-27 00:27:36          Grades PreK-2

  project_subject_categories            project_subject_subcategories  \
0             Math & Science  Applied Sciences, Health & Life Science

                               project_title  \
0  Engineering STEAM into the Primary Classroom

                                     project_essay_1  \
0  I have been fortunate enough to use the Fairy ...

                                     project_essay_2  \
0  My students come from a variety of backgrounds...

                                     project_essay_3  \
0  Each month I try to do several science or STEM...

                                     project_essay_4  \
0  It is challenging to develop high quality scie...

                            project_resource_summary  \
0  My students need STEM kits to learn critical s...

   teacher_number_of_previously_posted_projects  project_is_approved   price  \
0                                            53                    1  725.05

   quantity
   quantity
0         4
```

In [8]:

```
#Refer for documentation: https://www.geeksforgeeks.org/python-pandas-index-value_counts/
approved_not_approved=train_data['project_is_approved'].value_counts()
print(approved_not_approved)
print("*"*50)
approved_not_approved1=train_data['project_is_approved'].value_counts(normalize=True)
print("in percentage=",approved_not_approved1)
```

```
1    92706
0    16542
Name: project_is_approved, dtype: int64
**************************************************
in percentage= 1    0.848583
0    0.151417
Name: project_is_approved, dtype: float64
```

Feature Preprocessing

Preprocessing of project_subject_categories

In [9]:

```
#Refer ->https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
#Refer for documentation ->https://www.programiz.com/python-programming/methods/string/strip
categories = list(train_data['project_subject_categories'].values) #creating a list of  all the va
lues in project subject categories
clean_cat=[]
for i in categories: #taking each category at a time
    temp="" #creating a empty string
    for j in i.split(","): # splitting each word separated by a comma
        if 'The' in j.split():
            j=j.replace('The',"") #replacing the every occurence of "The" with ""
        j=j.replace(" ","") #replacing every white space with ""
        temp+=j.strip()+" " #removing all leading and trailing whitespaces and then adding a white
space at the end
        temp = temp.replace('&','') #replacing & with "_"
        temp=temp.lower()
    clean_cat.append(temp.strip())
    #showing the result
print(clean_cat[23])
```

```
mathscience
```

In [10]:

```
train_data['clean_categories']=clean_cat #creating a new column as clean_categories
train_data.drop(['project_subject_categories'], axis=1,inplace=True) #dropping the subject categor
y
```

In [11]:

```
# Counting number of words in a corpus/clean_categories
#Refer ->https://stackoverflow.com/questions/8139239/how-to-count-words-in-a-corpus-document
from collections import Counter
my_counter = Counter()
for word in train_data['clean_categories'].values:
    my_counter.update(word.split())

print(dict(my_counter)) #printing the dictionary
sortd=sorted(my_counter.items()) #with sorted function on dictionary it sorts in aplhabetical
order of value
print("="*50)
print(sortd)

# Refer -> sorting dictionary in python by value : https://www.geeksforgeeks.org/python-sort-pytho
# Refer -> sorting dictionary in python by value : https://www.geeksforgeeks.org/python-sort-pytho
n-dictionaries-by-key-or-value/
#https://www.geeksforgeeks.org/ways-sort-list-dictionaries-values-python-using-lambda-function/
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv:(kv[1] ,kv[0])))
```

```
{'mathscience': 41421, 'specialneeds': 13642, 'literacylanguage': 52239, 'appliedlearning': 12135,
'historycivics': 5914, 'musicarts': 10293, 'healthsports': 14223, 'warmth': 1388, 'carehunger':
1388}
==================================================
[('appliedlearning', 12135), ('carehunger', 1388), ('healthsports', 14223), ('historycivics', 5914
), ('literacylanguage', 52239), ('mathscience', 41421), ('musicarts', 10293), ('specialneeds',
13642), ('warmth', 1388)]
```

Preprocessing of project_subject_subcategories

In [12]:

```
#Refer ->https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
#Refer for documentation ->https://www.programiz.com/python-programming/methods/string/strip
subcategories = list(train_data['project_subject_subcategories'].values) #creating a list of  all
the values in project subject categories
clean_subcat=[]
for i in subcategories: #taking each category at a time
    temp="" #creating a empty string
    for j in i.split(","): # splitting each word separated by a comma
        if 'The' in j.split():
            j=j.replace('The',"") #replacing the every occurence of "The" with ""
        j=j.replace(" ","") #replacing every white space with ""
        temp+=j.strip()+" " #removing all leading and trailing whitespaces and then adding a white
space at the end
        temp = temp.replace('&','') #replacing & with "_"
        temp=temp.lower()
    clean_subcat.append(temp.strip())
    #showing the result
print(clean_subcat[24])
```

```
specialneeds
```

In [13]:

```
train_data['clean_subcategories']=clean_subcat #creating a new column as clean_categories
train_data.drop(['project_subject_subcategories'], axis=1,inplace=True) #dropping the subject cate
gory
```

In [14]:

```
# Counting number of words in a corpus/clean_categories
#Refer ->https://stackoverflow.com/questions/8139239/how-to-count-words-in-a-corpus-document
from collections import Counter
my_counter1 = Counter()
for word in train_data['clean_subcategories'].values:
    my_counter1.update(word.split())

print(dict(my_counter1)) #printing the dictionary
sortd1=sorted(my_counter1.items()) #with sorted function on dictionary it sorts in aplhabetical
order of value
print("="*50)
print(sortd1)

# Refer -> sorting dictionary in python by value : https://www.geeksforgeeks.org/python-sort-pytho
n-dictionaries-by-key-or-value/
#https://www.geeksforgeeks.org/ways-sort-list-dictionaries-values-python-using-lambda-function/
subcat_dict = dict(my_counter1)
sorted_subcat_dict = dict(sorted(subcat_dict.items(), key=lambda kv:(kv[1] ,kv[0])))
```

```
{'appliedsciences': 10816, 'healthlifescience': 4235, 'specialneeds': 13642, 'literacy': 33700, 'e
arlydevelopment': 4254, 'mathematics': 28074, 'socialsciences': 1920, 'historygeography': 3171, 'e
sl': 4367, 'extracurricular': 810, 'visualarts': 6278, 'environmentalscience': 5591,
'literaturewriting': 22179, 'gymfitness': 4509, 'music': 3145, 'teamsports': 2192,
'performingarts': 1961, 'collegecareerprep': 2568, 'other': 2372, 'charactereducation': 2065,
'foreignlanguages': 890, 'healthwellness': 10234, 'civicsgovernment': 815, 'economics': 269,
'communityservice': 441, 'financialliteracy': 568, 'nutritioneducation': 1355,
'parentinvolvement': 677, 'warmth': 1388, 'carehunger': 1388}
==================================================
[('appliedsciences', 10816), ('carehunger', 1388), ('charactereducation', 2065),
('civicsgovernment', 815), ('collegecareerprep', 2568), ('communityservice', 441),
('earlydevelopment', 4254), ('economics', 269), ('environmentalscience', 5591), ('esl', 4367), ('e
xtracurricular', 810), ('financialliteracy', 568), ('foreignlanguages', 890), ('gymfitness',
4509), ('healthlifescience', 4235), ('healthwellness', 10234), ('historygeography', 3171),
('literacy', 33700), ('literaturewriting', 22179), ('mathematics', 28074), ('music', 3145),
('nutritioneducation', 1355), ('other', 2372), ('parentinvolvement', 677), ('performingarts', 1961
), ('socialsciences', 1920), ('specialneeds', 13642), ('teamsports', 2192), ('visualarts', 6278),
('warmth', 1388)]
```

Text Preprocessing

First we have to merge all the essay columns into a single column and then count the number of words in essay's of approved projects and essay's of rejected projects

In [15]:

```
# merge two column text dataframe: https://stackoverflow.com/questions/19377969/combine-two-column
s-of-text-in-dataframe-in-pandas-python
train_data["project_essay"] = train_data["project_essay_1"].map(str) +train_data["project_essay_2"]
.map(str)+train_data["project_essay_3"].map(str) +  train_data["project_essay_4"].map(str)
            #Here the .map(str) converts string to all the coulmns in project_eassy_1/2/3/4
print(train_data['project_essay'].head(3))
```

```
0    I have been fortunate enough to use the Fairy ...
1    Imagine being 8-9 years old. You're in your th...
2    Having a class of 24 students comes with diver...
Name: project_essay, dtype: object
```

Essay Text

In [16]:

```
# 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 [17]:

```
s=set(stopwords.words('english'))
print(s)
```

```
{'we', 'on', 'off', 'just', 'any', 't', 'did', 'during', 'yourselves', 'be', 'does', 'they',
"needn't", 'no', 'ourselves', 'into', 'been', 'll', 'once', 'again', "aren't", "wasn't",
'herself', 'it', 'against', 'is', 'have', "didn't", 'after', 'very', 'below', 'hadn', 'with',
'if', 'couldn', 'my', 'were', 'o', 'before', 'such', 's', 'the', 'hasn', 'can', 'i', 'through', 'o
wn', 'and', "doesn't", 'do', 'mustn', 'between', 'until', 'ma', "isn't", 'which', "you've", 'its',
'in', 'didn', 'had', 'he', "hasn't", "it's", 'our', "that'll", 'me', 'under', 'too', "mightn't", '
his', "she's", 'these', 'their', 'over', "mustn't", 'ours', 'shouldn', 'few', 'haven', 'wouldn', '
above', 'only', 'how', 'she', 'same', 'are', 'mightn', 'themselves', 'of', 'for', 'doesn', 'wasn',
'myself', 'here', 'them', 'by', 'd', 'him', 'while', 'having', 'should', 'but', 'don', 'weren', 'm
'myself', 'here', 'them', 'by', 'd', 'him', 'while', 'having', 'should', 'but', 'don', 'weren', 'm
ore', 'now', 'ain', 'at', 'or', 'as', 'this', 'when', 'some', "weren't", 'am', "should've",
"don't", 'won', 'your', 'who', "won't", 'y', "you'll", "haven't", 'what', 'because', 'doing', 'her
s', 'both', 'a', 'where', "hadn't", 'has', 'was', 'aren', 'down', 'then', 'isn', 'yourself', 'whom
', 'himself', 're', 'shan', 'all', 'needn', "shan't", 'an', 'itself', 'there', 'not', "you're", 't
hose', 'theirs', 'up', 'why', "you'd", 'that', 'each', "shouldn't", 'so', 'will', 'm', 'out',
'most', 'you', 'nor', "couldn't", 'about', 'from', 'being', 'further', 'other', 'her', 'yours', 't
o', 've', 'than', "wouldn't"}
```

In [18]:

```
#Combining all the above statments to transform our text in a clean text
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(train_data['project_essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    sent=sent.lower()
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in s)
    preprocessed_essays.append(sent.strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:53<00:00, 2040.43it/s]
```

In [19]:

```
#printing the text after preprocessing
preprocessed_essays[0]
```

Out[19]:

```
'fortunate enough use fairy tale stem kits classroom well stem journals students really enjoyed wo
uld love implement lakeshore stem kits classroom next school year provide excellent engaging stem
lessons students come variety backgrounds including language socioeconomic status many lot experie
nce science engineering kits give materials provide exciting opportunities students month try seve
ral science stem steam projects would use kits robot help guide science instruction engaging
meaningful ways adapt kits current language arts pacing guide already teach material kits like tal
l tales paul bunyan johnny appleseed following units taught next school year implement kits
magnets motion sink vs float robots often get units know teaching right way using right materials
kits give additional ideas strategies lessons prepare students science challenging develop high qu
ality science activities kits give materials need provide students science activities go along
curriculum classroom although things like magnets classroom know use effectively kits provide righ
t amount materials show use appropriate way'
```

In [20]:

```
train_data['preprocessed_essays']=preprocessed_essays
train_data.drop(['project_essay'], axis=1,inplace=True)
```

Project title text

In [21]:

```
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for title in tqdm(train_data['project_title'].values):
    test1 = decontracted(title)
    test1 = test1.replace('\\r', ' ')
    test1 = test1.replace('\\"', ' ')
    test1 = test1.replace('\\n', ' ')
    test1 = re.sub('[^A-Za-z0-9]+', ' ', test1)
    test1=test1.lower()
    # https://gist.github.com/sebleier/554280
    test1 = ' '.join(e for e in test1.split() if e not in s)
    preprocessed_title.append(test1.strip())
```

```
100%|████████████████████████████████████████████████████████████████████████| 109248/109248
[00:05<00:00, 20000.11it/s]
```

In [22]:

```
preprocessed_title[0]
```

Out[22]:

```
'engineering steam primary classroom'
```

In [23]:

```
train_data['preprocessed_title']=preprocessed_title
train_data.drop(['project_title'], axis=1,inplace=True)
```

Category Preprocessing

Teacher Prefix

In [24]:

```
from tqdm import tqdm
import string
preprocessed_prefix=[]
for prefix in tqdm(train_data['teacher_prefix'].values):
    test=str(prefix).strip(".")
    test=test.lower()
    preprocessed_prefix.append(test)
```

```
100%|███████████████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 361268.85it/s]
```

In [25]:

```
preprocessed_prefix[3]
```

Out[25]:

```
'mrs'
```

In [26]:

```
train_data['preprocessed_prefix']=preprocessed_prefix
#train_data.drop(['teacher_prefix'], axis=1,inplace=True)
```

Grade Category

In [27]:

```
preprocessed_grade=[]
for grade in tqdm(train_data['project_grade_category'].values):
    grade=grade.strip(" ")
    grade=grade.replace(" ", "_")
    grade=grade.replace("-","_")
    preprocessed_grade.append(grade)
```

```
100%|███████████████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 304879.75it/s]
```

In [28]:

```
preprocessed_grade[0:5]
```

```
preprocessed_grade[0:5]
```

Out[28]:

```
['Grades_PreK_2', 'Grades_3_5', 'Grades_PreK_2', 'Grades_PreK_2', 'Grades_3_5']
```

In [29]:

```
train_data['preprocessed_grade']=preprocessed_grade
train_data.drop(['project_grade_category'], axis=1,inplace=True)
```

project_resource_summary

In [30]:

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

```
100%|████████████████████████████████████████████████████████████| 109248/109248
[00:08<00:00, 13131.92it/s]
```

In [31]:

```
preprocessed_resource[0:5]
```

Out[31]:

```
['students need stem kits learn critical science engineering skills kits focus important science c
oncepts robot works engineering skills',
 'students need boogie boards quiet sensory breaks putty sensory input focus',
 'students need mobile listening center able enhance learning',
 'students need flexible seating classroom choose comfortable learn best',
 'students need copies new york times best seller wonder book okay think deeply compare contrast s
tructures']
```

In [32]:

```
train_data['preprocessed_resource']=preprocessed_resource
train_data.drop(['project_resource_summary'], axis=1,inplace=True)
```

In [33]:

```
X=train_data.drop(columns=['id',"teacher_id","Date",'project_essay_1','project_essay_2','project_es
say_3','project_essay_4'])
```

In [34]:

```
y=X['project_is_approved']
```

In [35]:

```
X=X.drop(columns=['project_is_approved','teacher_prefix'])
```

In [36]:

```
print(X.head(3))
```

```
   Unnamed: 0 school_state  teacher_number_of_previously_posted_projects  \
0        8393           CA                                            53   
1       37728           UT                                             4   
2       74477           CA                                            10   

    price  quantity  clean_categories                clean_subcategories  \
0  725.05         4       mathscience  appliedsciences healthlifescience   
1  213.03         8      specialneeds                       specialneeds   
2  329.00         1  literacylanguage                           literacy   

                                preprocessed_essays  \
0  fortunate enough use fairy tale stem kits clas...   
1  imagine 8 9 years old third grade classroom se...   
2  class 24 students comes diverse learners stude...   

                        preprocessed_title preprocessed_prefix  \
0      engineering steam primary classroom                 mrs   
1                      sensory tools focus                  ms   
2  mobile learning mobile listening center                 mrs   

  preprocessed_grade                              preprocessed_resource  
0      Grades_PreK_2  students need stem kits learn critical science...  
1         Grades_3_5  students need boogie boards quiet sensory brea...  
2      Grades_PreK_2  students need mobile listening center able enh...  
```

In [37]:

```python
print(X.shape)
print("="*50)
print(y.shape)
```

```
(109248, 12)
==================================================
(109248,)
```

Data Splitting into train,cv and test

In [38]:

```python
# ============================== loading libraries ===========================================
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import model_selection
# =============================================================================================
```

In [39]:

```python
# split the data set into train and test
#how to stratify using knn->https://stackoverflow.com/questions/34842405/parameter-stratify-from-m
ethod-train-test-split-scikit-learn
X_1, X_test, y_1, y_test =model_selection.train_test_split(X,y, test_size=0.33, random_state=5,stra
tify= y)#random spliiting of data into test and train
```

In [40]:

```python
X_train, X_cv, y_train, y_cv = train_test_split(X_1, y_1, test_size=0.33,random_state=5,stratify= y
_1) # this is random splitting of train data into train anc cross-validation
```

In [41]:

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

print("="*100)
```

```
(49041, 12) (49041,)
(24155, 12) (24155,)
(36052, 12) (36052,)
====================================================================================================
```

Vectorization

One-Hot encoding of categorical feature

Category Feature

In [42]:

```
#instantiate
vectorizer_cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True) #creating vocabulary
#fitting
vectorizer_cat.fit(X_train['clean_categories'].values) #learning from the train data
print(vectorizer_cat.get_feature_names())
print('='*50)
#Transform
categories_ohe_train=vectorizer_cat.transform(X_train['clean_categories'].values)#applying learned parameters to train,test and cv values
print("Shape of train data after one hot encoding",categories_ohe_train.shape)
print("train data after one hot encoding",categories_ohe_train[0:5, :])
categories_ohe_cv=vectorizer_cat.transform(X_cv['clean_categories'].values)
print('='*50)
print("Shape of CV data after one hot encoding",categories_ohe_cv.shape)
print("CV data after one hot encoding",categories_ohe_cv[0:5, :])
categories_ohe_test=vectorizer_cat.transform(X_test['clean_categories'].values)
print('='*50)
print("Shape of test data after one hot encoding",categories_ohe_test.shape)
print("test data after one hot encoding",categories_ohe_test[0:5, :])
```

```
['carehunger', 'warmth', 'historycivics', 'musicarts', 'appliedlearning', 'specialneeds',
'healthsports', 'mathscience', 'literacylanguage']
==================================================
Shape of train data after one hot encoding (49041, 9)
train data after one hot encoding   (0, 8)	1
  (1, 8)	1
  (2, 0)	1
  (2, 1)	1
  (3, 7)	1
  (4, 2)	1
==================================================
Shape of CV data after one hot encoding (24155, 9)
CV data after one hot encoding   (0, 7)	1
  (1, 2)	1
  (1, 4)	1
  (2, 8)	1
  (3, 7)	1
  (4, 6)	1
==================================================
Shape of test data after one hot encoding (36052, 9)
test data after one hot encoding   (0, 5)	1
  (0, 8)	1
  (1, 7)	1
  (2, 4)	1
  (2, 8)	1
  (3, 7)	1
  (4, 8)	1
```

Sub-Category feature

In [43]:

```
vectorizer_sub_cat = CountVectorizer(vocabulary=list(sorted_subcat_dict.keys()), lowercase=False,
binary=True)
vectorizer_sub_cat.fit(X_train['clean_subcategories'].values)
print(vectorizer_sub_cat.get_feature_names())
print('='*50)
subcategories_ohe_train=vectorizer_sub_cat.transform(X_train['clean_subcategories'].values)#applyin
g learned parameters to train,test and cv values
print("Shape of train data after one hot encoding",subcategories_ohe_train.shape)
print("train data after one hot encoding",subcategories_ohe_train[0:5,:])
subcategories_ohe_cv=vectorizer_sub_cat.transform(X_cv['clean_subcategories'].values)
print('='*50)
print("Shape of CV data after one hot encoding",subcategories_ohe_cv.shape)
print("CV data after one hot encoding",subcategories_ohe_cv[0:5,:])
subcategories_ohe_test=vectorizer_sub_cat.transform(X_test['clean_subcategories'].values)
print('='*50)
print("Shape of test data after one hot encoding",subcategories_ohe_test.shape)
print("test data after one hot encoding",subcategories_ohe_test[0:5,:])
```

```
['economics', 'communityservice', 'financialliteracy', 'parentinvolvement', 'extracurricular',
'civicsgovernment', 'foreignlanguages', 'nutritioneducation', 'carehunger', 'warmth',
'socialsciences', 'performingarts', 'charactereducation', 'teamsports', 'other',
'collegecareerprep', 'music', 'historygeography', 'healthlifescience', 'earlydevelopment', 'esl',
'gymfitness', 'environmentalscience', 'visualarts', 'healthwellness', 'appliedsciences',
'specialneeds', 'literaturewriting', 'mathematics', 'literacy']
==================================================
Shape of train data after one hot encoding (49041, 30)
train data after one hot encoding   (0, 27) 1
  (1, 27) 1
  (2, 8) 1
  (2, 9) 1
  (3, 22) 1
  (4, 2) 1
  (4, 10) 1
==================================================
Shape of CV data after one hot encoding (24155, 30)
CV data after one hot encoding   (0, 22) 1
  (1, 15) 1
  (1, 17) 1
  (2, 29) 1
  (3, 28) 1
  (4, 13) 1
  (4, 24) 1
==================================================
Shape of test data after one hot encoding (36052, 30)
test data after one hot encoding   (0, 6) 1
  (0, 26) 1
  (1, 22) 1
  (1, 25) 1
  (2, 19) 1
  (2, 29) 1
  (3, 22) 1
  (4, 29) 1
```

School-State feature

In [44]:

```
#counting number of words in the project grade category and then coverting into dictionary
from collections import Counter
my_counter=Counter()
for state in train_data['school_state'].values:
    my_counter.update(state.split())

#Converting to dictionary
school_state_dict=dict(my_counter)
#sorting
sorted_school_state_dict=dict(sorted(school_state_dict.items(),key=lambda kv:(kv[1],kv[0])))
```

In [45]:

```
vectorizer_school = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lowercase=Fa
lse, binary=True)
vectorizer_school.fit(X_train['school_state'].values)
print(vectorizer_school.get_feature_names())
print('='*50)
state_ohe_train=vectorizer_school.transform(X_train['school_state'].values)#applying learned
parameters to train,test and cv values
print("Shape of train data after one hot encoding",state_ohe_train.shape)
print("train data after one hot encoding",state_ohe_train[0:5,:])
state_ohe_cv=vectorizer_school.transform(X_cv['school_state'].values)
print('='*50)
print("Shape of CV data after one hot encoding",state_ohe_cv.shape)
print("CV data after one hot encoding",state_ohe_cv[0:5,:])
state_ohe_test=vectorizer_school.transform(X_test['school_state'].values)
print('='*50)
print("Shape of test data after one hot encoding",state_ohe_test.shape)
print("test data after one hot encoding",state_ohe_test[0:5,:])
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'I
A', 'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ',
'NJ', 'OK', 'WA', 'MA', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX
', 'CA']
==================================================
Shape of train data after one hot encoding (49041, 51)
train data after one hot encoding   (0, 44) 1
  (1, 49) 1
  (2, 12) 1
  (3, 50) 1
  (4, 32) 1
==================================================
Shape of CV data after one hot encoding (24155, 51)
CV data after one hot encoding   (0, 23) 1
  (1, 22) 1
  (2, 42) 1
  (3, 40) 1
  (4, 41) 1
==================================================
Shape of test data after one hot encoding (36052, 51)
test data after one hot encoding   (0, 45) 1
  (1, 47) 1
  (2, 30) 1
  (3, 25) 1
  (4, 13) 1
```

Project_Grade feature

In [46]:

```
from collections import Counter
my_counter1 = Counter()
for word in train_data['preprocessed_grade'].values:
    my_counter1.update(word.split())

#converting to dictionary
project_grade_dict=dict(my_counter1)
#Now sorting the dictionary
sorted_project_grade_dict = dict(sorted(project_grade_dict.items(), key=lambda kv:(kv[1] ,kv[0])))
print(sorted_project_grade_dict)
```

```
{'Grades_9_12': 10963, 'Grades_6_8': 16923, 'Grades_3_5': 37137, 'Grades_PreK_2': 44225}
```

In [47]:

```
#How to remove values from a dictionary in python-> https://thispointer.com/different-ways-to-remo
ve-a-key-from-dictionary-in-python/
if 'Grades' in sorted_project_grade_dict:
    del sorted_project_grade_dict['Grades']

print("Updated Dictionary :" , sorted_project_grade_dict)
```

```
Updated Dictionary : {'Grades_9_12': 10963, 'Grades_6_8': 16923, 'Grades_3_5': 37137,
'Grades_PreK_2': 44225}
```

In [48]:

```
vectorizer_grade = CountVectorizer(vocabulary=list(sorted_project_grade_dict.keys()), lowercase=False, binary=True)
vectorizer_grade.fit(X_train['preprocessed_grade'].values)
print(vectorizer_grade.get_feature_names())
print('='*50)
grade_ohe_train=vectorizer_grade.transform(X_train['preprocessed_grade'].values)#applying learned parameters to train,test and cv values
print("Shape of train data after one hot encoding",grade_ohe_train.shape)
print("train data after one hot encoding",grade_ohe_train[0:5,:])
grade_ohe_cv=vectorizer_grade.transform(X_cv['preprocessed_grade'].values)
print('='*50)
print("Shape of CV data after one hot encoding",grade_ohe_cv.shape)
print("cv data after one hot encoding",grade_ohe_cv[0:5,:])
grade_ohe_test=vectorizer_grade.transform(X_test['preprocessed_grade'].values)
print('='*50)
print("Shape of test data after one hot encoding",grade_ohe_test.shape)
print("test data after one hot encoding",grade_ohe_test[0:5,:])
```

```
['Grades_9_12', 'Grades_6_8', 'Grades_3_5', 'Grades_PreK_2']
==================================================
Shape of train data after one hot encoding (49041, 4)
train data after one hot encoding   (0, 0) 1
  (1, 2) 1
  (2, 2) 1
  (3, 3) 1
  (4, 1) 1
==================================================
Shape of CV data after one hot encoding (24155, 4)
cv data after one hot encoding   (0, 2) 1
  (1, 0) 1
  (2, 3) 1
  (3, 3) 1
  (4, 0) 1
==================================================
Shape of test data after one hot encoding (36052, 4)
test data after one hot encoding   (0, 1) 1
  (1, 2) 1
  (2, 3) 1
  (3, 3) 1
  (4, 3) 1
```

Teacher-Prefix feature

In [49]:

```
train_data['preprocessed_prefix']= train_data['preprocessed_prefix'].fillna('missing')
print("="*50)
print(train_data['preprocessed_prefix'].value_counts())
```

```
==================================================
mrs        57269
ms         38955
mr         10648
teacher     2360
dr            13
nan            3
Name: preprocessed_prefix, dtype: int64
```

In [50]:

```
from collections import Counter
my_counter1 = Counter()
for word in train_data['preprocessed_prefix'].values:
    my_counter1.update(word.split())

#converting to dictionary
```

```
teacher_prefix_dict=dict(my_counter1)
#Now sorting the dictionary
sorted_teacher_prefix_grade_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda kv:(kv[1] ,k
v[0])))
print(sorted_teacher_prefix_grade_dict)
```

```
{'nan': 3, 'dr': 13, 'teacher': 2360, 'mr': 10648, 'ms': 38955, 'mrs': 57269}
```

In [51]:

```
#to counter error: np.nan is an invalid document, expected byte or unicode string.
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-is
-an-invalid-document

vectorizer_prefix = CountVectorizer(vocabulary=list(sorted_teacher_prefix_grade_dict.keys()), lowe
rcase=False, binary=True)
vectorizer_prefix.fit(X_train['preprocessed_prefix'].values.astype('U'))
print(vectorizer_prefix.get_feature_names())
print('='*50)
prefix_ohe_train=vectorizer_prefix.transform(X_train['preprocessed_prefix'].values.astype('U'))#app
lying learned parameters to train,test and cv values
print("Shape of train data after one hot encoding",prefix_ohe_train.shape)
print("train data after one hot encoding",prefix_ohe_train[0:5,:])
prefix_ohe_cv=vectorizer_prefix.transform(X_cv['preprocessed_prefix'].values.astype('U'))
print('='*50)
print("Shape of CV data after one hot encoding",prefix_ohe_cv.shape)
print("cv data after one hot encoding",prefix_ohe_cv[0:5,:])
prefix_ohe_test=vectorizer_prefix.transform(X_test['preprocessed_prefix'].values.astype('U'))
print('='*50)
print("Shape of test data after one hot encoding",prefix_ohe_test.shape)
print("test data after one hot encoding",prefix_ohe_test[0:5,:])
```

```
['nan', 'dr', 'teacher', 'mr', 'ms', 'mrs']
==================================================
Shape of train data after one hot encoding (49041, 6)
train data after one hot encoding   (0, 3)  1
  (1, 5)  1
  (2, 4)  1
  (3, 5)  1
  (4, 5)  1
==================================================
Shape of CV data after one hot encoding (24155, 6)
cv data after one hot encoding   (0, 5)  1
  (1, 3)  1
  (2, 4)  1
  (3, 3)  1
  (4, 4)  1
==================================================
Shape of test data after one hot encoding (36052, 6)
test data after one hot encoding   (0, 5)  1
  (1, 4)  1
  (2, 4)  1
  (3, 4)  1
  (4, 4)  1
```

Normalizing Numerical Features

Price feature

In [52]:

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


price_train=price_scalar.transform(X_train['price'].values.reshape(1,-1))
print("Shape of price train data after normalization",price_train.shape)
print("price train data after normalization",price_train[0:1])
print("="*50)
price_cv=price_scalar.transform(X_cv['price'].values.reshape(1, -1))
```

```
print("Shape of price CV data after normalization",price_cv.shape)
print("price cv data after normalization",price_cv[0:1])
print("="*50)
price_test=price_scalar.transform(X_test['price'].values.reshape(1 ,-1))
print("Shape of price test data after normalization",price_test.shape)
print("price test data after normalization",price_test[0:1])
```

```
Shape of price train data after normalization (1, 49041)
price train data after normalization [[0.00048689 0.00031167 0.00240483 ... 0.00253076 0.00389578
0.00197083]]
==================================================
Shape of price CV data after normalization (1, 24155)
price cv data after normalization [[0.0012673  0.02562521 0.00240981 ... 0.0016012  0.00835091
0.00321995]]
==================================================
Shape of price test data after normalization (1, 36052)
price test data after normalization [[0.01552693 0.00111198 0.00429254 ... 0.00451968 0.00102011
0.01211403]]
```

In [53]:

```
# Reshaping Again
price_train=price_train.reshape(-1,1)
print("after reshape",price_train.shape)
price_cv=price_cv.reshape(-1,1)
print("after reshape",price_cv.shape)
price_test=price_test.reshape(-1,1)
print("after reshape",price_test.shape)
```

```
after reshape (49041, 1)
after reshape (24155, 1)
after reshape (36052, 1)
```

Quantity Feature

In [54]:

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


quantity_train=quantity_scalar.transform(X_train['quantity'].values.reshape(1, -1))
print("Shape of quantity train data after normalization",quantity_train.shape)
print("quantity train data after normalization",quantity_train[0:1])
print("="*50)
quantity_cv=quantity_scalar.transform(X_cv['quantity'].values.reshape(1, -1))
print("Shape of quantity CV data after normalization",quantity_cv.shape)
print("quantity cv data after normalization",quantity_cv[0:1])
print("="*50)
quantity_test=quantity_scalar.transform(X_test['quantity'].values.reshape(1, -1))
print("Shape of quantity test data after normalization",quantity_test.shape)
print("quantity test data after normalization",quantity_test[0:1])
```

```
Shape of quantity train data after normalization (1, 49041)
quantity train data after normalization [[0.00210819 0.01503845 0.00520021 ... 0.00281093
0.00028109 0.00351366]]
==================================================
Shape of quantity CV data after normalization (1, 24155)
quantity cv data after normalization [[0.00206929 0.00020693 0.00124158 ... 0.00082772 0.00227622
0.00103465]]
==================================================
Shape of quantity test data after normalization (1, 36052)
quantity test data after normalization [[0.00017579 0.00193368 0.00070315 ... 0.00017579
0.00105473 0.00052737]]
```

In [55]:

```
# Reshaping Again
quantity_train=quantity_train.reshape(-1,1)
print("after reshape",price_train.shape)
print("after reshape",price_train.shape)
quantity_cv=quantity_cv.reshape(-1,1)
print("after reshape",price_cv.shape)
quantity_test=quantity_test.reshape(-1,1)
print("after reshape",price_test.shape)
```

```
after reshape (49041, 1)
after reshape (24155, 1)
after reshape (36052, 1)
```

Teacher number of previously posted projects feature

In [56]:

```
tnp_scalar = Normalizer()
tnp_scalar.fit(X_train["teacher_number_of_previously_posted_projects"].values.reshape(1,-1)) # find
ing the mean and standard deviation of this data

# Now standardize the data with above maen and variance.
tnp_train = tnp_scalar.transform(X_train["teacher_number_of_previously_posted_projects"].values.re
shape(1, -1))
print(tnp_train.shape)
print("train data after normalization",tnp_train[0:1])
print('='*50)
tnp_cv = tnp_scalar.transform(X_cv["teacher_number_of_previously_posted_projects"].values.reshape(
1, -1))
print(tnp_cv.shape)
print("cv data after normalization",tnp_cv[0:1])
print('='*50)
tnp_test =
tnp_scalar.transform(X_test["teacher_number_of_previously_posted_projects"].values.reshape(1, -1))
print(tnp_test.shape)
print("test data after normalization",tnp_test[0:1])
```

```
(1, 49041)
train data after normalization [[0.         0.00090827 0.00105964 ... 0.00015138 0.         0.
]]
==================================================
(1, 24155)
cv data after normalization [[0.0006398  0.         0.         ... 0.00085307 0.00191941 0.
]]
==================================================
(1, 36052)
test data after normalization [[0.00052822 0.         0.0026411  ... 0.         0.
0.00052822]]
```

In [57]:

```
# Reshaping Again
tnp_train=tnp_train.reshape(-1,1)
print("after reshape",price_train.shape)
tnp_cv=tnp_cv.reshape(-1,1)
print("after reshape",price_cv.shape)
tnp_test=tnp_test.reshape(-1,1)
print("after reshape",price_test.shape)
```

```
after reshape (49041, 1)
after reshape (24155, 1)
after reshape (36052, 1)
```

Vectorizing Test Data

Bag of words(BoW)

Preprocessed Essay

In [58]:

```
model_essay_bow =  CountVectorizer(min_df=10,ngram_range=(1,2),max_features=5000)
model_essay_bow.fit(X_train["preprocessed_essays"])


train_bow_essay = model_essay_bow.transform(X_train["preprocessed_essays"])
print("Shape of matrix ",train_bow_essay.shape)
print("="*50)
cv_bow_essay=model_essay_bow.transform(X_cv["preprocessed_essays"]) #BoW of CV
print("Shape of matrix ",cv_bow_essay.shape)
print("="*50)
test_bow_essay = model_essay_bow.transform(X_test["preprocessed_essays"]) #BoW of Test
print("Shape of matrix ",test_bow_essay.shape)
```

```
Shape of matrix  (49041, 5000)
==================================================
Shape of matrix  (24155, 5000)
==================================================
Shape of matrix  (36052, 5000)
```

In [59]:

```
pre_essay_tokens=model_essay_bow.get_feature_names()
len(pre_essay_tokens)
```

Out[59]:

```
5000
```

In [60]:

```
df_pre=pd.Series(pre_essay_tokens)
type(df_pre)
```

Out[60]:

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

In [61]:

```
#printing the first 10 and last 10 tokens of a preprocessed essays
print(df_pre.head(10))
print(df_pre.tail(10))
```

```
0              000
1               10
2              100
3         100 free
4      100 percent
5     100 students
6               11
7               12
8             12th
9       12th grade
dtype: object
4990           young age
4991      young children
4992      young learners
4993         young minds
4994        young people
4995      young students
4996             younger
4997    younger students
4998               youth
4999                zone
dtype: object
```

Preprocessed Title

In [62]:

```
model_title_bow =  CountVectorizer()
model_title_bow.fit(X_train["preprocessed_title"])
train_bow_title = model_title_bow.transform(X_train["preprocessed_title"])
print("Shape of matrix ",train_bow_title.shape)
print("="*50)
cv_bow_title=model_title_bow.transform(X_cv["preprocessed_title"]) #BoW of test
print("Shape of matrix ",cv_bow_title.shape)
print("="*50)
test_bow_title = model_title_bow.transform(X_test["preprocessed_title"]) #BoW of Cross Validation
print("Shape of matrix ",test_bow_title.shape)
```

```
Shape of matrix  (49041, 11693)
==================================================
Shape of matrix  (24155, 11693)
==================================================
Shape of matrix  (36052, 11693)
```

In [63]:

```
df_title=pd.Series(model_title_bow.get_feature_names())
type(df_title)
```

Out[63]:

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

In [64]:

```
#printing the first 10 and last 10 tokens of a preprocessed essays
print(df_title.head(10))
print(df_title.tail(10))
```

```
0    000
1     02
2     03
3     04
4     05
5     06
6     07
7     09
8     0n
9     0s
dtype: object
11683      zombie
11684        zone
11685       zones
11686         zoo
11687        zoom
11688     zooming
11689          zu
11690        zuma
11691       zumba
11692    zwieback
dtype: object
```

Tf-idf

Preprocessed Essay

In [65]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
model_essay_tfidf = TfidfVectorizer(min_df=10,ngram_range=(1,2),max_features=5000) #df tells us tha
t we will only consider those words which is present atleast in 10 documents
model_essay_tfidf.fit(X_train["preprocessed_essays"])
train_tfidf_essay = model_essay_tfidf.transform(X_train["preprocessed_essays"])
print("Shape of matrix ",train_tfidf_essay.shape)
print("="*50)
cv_tfidf_essay=model_essay_tfidf.transform(X_cv["preprocessed_essays"]) #BoW of test
```

```
print("Shape of matrix ",cv_tfidf_essay.shape)
print("="*50)
test_tfidf_essay= model_essay_tfidf.transform(X_test["preprocessed_essays"]) #BoW of Cross
Validation
print("Shape of matrix ",test_tfidf_essay.shape)
```

```
Shape of matrix  (49041, 5000)
==================================================
Shape of matrix  (24155, 5000)
==================================================
Shape of matrix  (36052, 5000)
```

In [66]:

```
df_pre_tfidf=pd.Series(model_essay_tfidf.get_feature_names())
type(df_pre_tfidf)
```

Out[66]:

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

In [67]:

```
#printing the first 10 and last 10 tokens of a preprocessed essays
print(df_pre_tfidf.head(10))
print(df_pre_tfidf.tail(10))
```

```
0              000
1               10
2              100
3         100 free
4      100 percent
5     100 students
6               11
7               12
8             12th
9       12th grade
dtype: object
4990           young age
4991      young children
4992      young learners
4993         young minds
4994        young people
4995      young students
4996             younger
4997    younger students
4998               youth
4999                zone
dtype: object
```

Preprocessed Title

In [68]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
model_title_tfidf = TfidfVectorizer()
model_title_tfidf.fit(X_train["preprocessed_title"])
train_tfidf_title = model_title_tfidf.transform(X_train["preprocessed_title"])
print("Shape of matrix ",train_tfidf_title.shape)
print("="*50)
cv_tfidf_title=model_title_tfidf.transform(X_cv["preprocessed_title"])
print("Shape of matrix ",cv_tfidf_title.shape)
print("="*50)
test_tfidf_title= model_title_tfidf.transform(X_test["preprocessed_title"])
print("Shape of matrix ",test_tfidf_title.shape)
```

```
Shape of matrix  (49041, 11693)
==================================================
Shape of matrix  (24155, 11693)
==================================================
Shape of matrix  (36052, 11693)
Shape of matrix  (50052, 11693)
```

In [69]:

```
df_title_tfidf=pd.Series(model_title_tfidf.get_feature_names())
type(df_title_tfidf)
```

Out[69]:

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

In [70]:

```
#printing the first 10 and last 10 tokens of a preprocessed essays
print(df_title_tfidf.head(10))
print(df_title_tfidf.tail(10))
```

```
0     000
1      02
2      03
3      04
4      05
5      06
6      07
7      09
8      0n
9      0s
dtype: object
11683      zombie
11684        zone
11685       zones
11686         zoo
11687        zoom
11688     zooming
11689          zu
11690        zuma
11691       zumba
11692    zwieback
dtype: object
```

Average word2vector(avg w2v)

In [71]:

```
#https://stackoverflow.com/questions/49083826/get-trouble-to-load-glove-840b-300d-vector
import numpy as np
from tqdm import tqdm
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.asarray(splitLine[1:], dtype='float32')
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
```

In [72]:

```
model = loadGloveModel('glove.840B.300d.txt')
```

```
Loading Glove Model
```

```
2196017it [10:37, 3442.25it/s]
```

```
Done. 2196016  words loaded!
```

In [73]:

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

In [74]:

```
print("all the words in the corpus", len(words))
words = set(words)
print("the unique words in the corpus", len(words))

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

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

```
all the words in the corpus 6693865
the unique words in the corpus 41189
The number of words that are present in both glove vectors and our corpus 36109 ( 87.667 %)
word 2 vec length 36109
```

In [75]:

```
import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(train_words_corpus, f) # save training datasets into a pickle file for machine
learning
```

In [76]:

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

Train Essays

In [77]:

```
# average Word2Vec
# compute average word2vec for each test data

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

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

```
100%|██████████████████████████████████████████████████████████████████████████████| 49041/49041 [01:
11<00:00, 688.38it/s]
```

```
49041
300
```

Cross-Validation Essays

In [78]:

```
# average Word2Vec
# compute average word2vec for each CV data

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

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

```
100%|██████████████████████████████████████████████████████████████████████████████| 24155/24155 [00:
35<00:00, 689.51it/s]
```

```
24155
300
```

Test Essays

In [79]:

```
# average Word2Vec
# compute average word2vec for each test data

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

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

```
100%|██████████████████████████████████████████████████████████████████████████████| 36052/36052 [00:
48<00:00, 750.78it/s]
```

```
36052
300
```

Train Titles

In [80]:

```
# average Word2Vec
# compute average word2vec for each training data

from tqdm import tqdm
avg_w2v_vectors_title_train = []; # the avg-w2v for each essays is stored in this list
```

```
for sentence in tqdm(X_train["preprocessed_title"]): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a esssay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_train.append(vector)

print(len(avg_w2v_vectors_title_train))
print(len(avg_w2v_vectors_title_train[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 49041/49041
[00:03<00:00, 15310.60it/s]
```

```
49041
300
```

Cross-Validation Ttiles

In [81]:

```
# average Word2Vec
# compute average word2vec for each CV data

from tqdm import tqdm
avg_w2v_vectors_title_cv = []; # the avg-w2v for each essays is stored in this list
for sentence in tqdm(X_cv["preprocessed_title"]): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a esssay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_cv.append(vector)

print(len(avg_w2v_vectors_title_cv))
print(len(avg_w2v_vectors_title_cv[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 24155/24155
[00:01<00:00, 13614.52it/s]
```

```
24155
300
```

Test Titles

In [82]:

```
# average Word2Vec
# compute average word2vec for each test data

from tqdm import tqdm
avg_w2v_vectors_title_test = []; # the avg-w2v for each essays is stored in this list
for sentence in tqdm(X_test["preprocessed_title"]): # for each essay
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a esssay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_test.append(vector)

print(len(avg_w2v_vectors_title_test))
print(len(avg_w2v_vectors_title_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 36052/36052
[00:02<00:00, 14636.03it/s]
```

```
36052
300
```

Tf-idf weighted W2V

Using Pretrained Model for finding the tf-idf weighted word2vec

Train Essays

In [83]:

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

In [84]:

```
# compute average word2vec for Training Data
from tqdm import tqdm
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence
for sentence in tqdm(X_train["preprocessed_essays"]): # for each 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
    for word in sentence.split(): # for each word in a sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf 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_train.append(vector)

print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████| 49041/49041 [05:
13<00:00, 156.50it/s]
```

```
49041
300
```

Cross-Validation Essays

In [85]:

```
# compute average word2vec for Cross Validation data
from tqdm import tqdm
tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence
for sentence in tqdm(X_cv["preprocessed_essays"]): # for each 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
    for word in sentence.split(): # for each word in a sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
```

```
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf 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_cv.append(vector)

print(len(tfidf_w2v_vectors_cv))
print(len(tfidf_w2v_vectors_cv[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 24155/24155 [03:
21<00:00, 119.77it/s]
```

```
24155
300
```

Test Essays

In [86]:

```
# compute average word2vec for test data
from tqdm import tqdm
tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence
for sentence in tqdm(X_test["preprocessed_essays"]): # for each 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
    for word in sentence.split(): # for each word in a sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf 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%|██████████████████████████████████████████████████████████████████████████| 36052/36052 [04:
44<00:00, 126.82it/s]
```

```
36052
300
```

Train Titles

In [87]:

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

In [88]:

```
# compute average word2vec for Training Data
from tqdm import tqdm
tfidf_w2v_vectors_title_train = []; # the avg-w2v for each sentence
for sentence in tqdm(X_train["preprocessed_title"]): # for each 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
    for word in sentence.split(): # for each word in a sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf 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_train.append(vector)

print(len( tfidf_w2v_vectors_title_train))
print(len( tfidf_w2v_vectors_title_train[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 49041/49041
[00:06<00:00, 7958.09it/s]
```

```
49041
300
```

Cross-Validation Titles

In [89]:

```
# compute average word2vec for Cross-Validation Data
from tqdm import tqdm
tfidf_w2v_vectors_title_cv = []; # the avg-w2v for each sentence
for sentence in tqdm(X_cv["preprocessed_title"]): # for each 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
    for word in sentence.split(): # for each word in a sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf 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_cv.append(vector)

print(len( tfidf_w2v_vectors_title_cv))
print(len( tfidf_w2v_vectors_title_cv[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 24155/24155
[00:02<00:00, 8160.76it/s]
```

```
24155
300
```

Test titles

In [90]:

```
# compute average word2vec for Test Data
from tqdm import tqdm
tfidf_w2v_vectors_title_test = []; # the avg-w2v for each sentence
for sentence in tqdm(X_test["preprocessed_title"]): # for each 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
    for word in sentence.split(): # for each word in a sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf 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%|██████████████████████████████████████████████████████████████████████████| 36052/36052
[00:04<00:00, 7633.44it/s]
```

```
36052
300
```

Applying Logistic Regression

Set 1: Categorical Features,Numerical Features+Preprocessed Essay(BOW with unigram and bigram and max features 5000 and min df=10)+Preprocessed Title(BOW)

In [91]:

```
from scipy.sparse import hstack
X_tr=hstack((categories_ohe_train,subcategories_ohe_train,state_ohe_train,grade_ohe_train,prefix_oh
e_train,price_train,quantity_train,tnp_train,train_bow_essay,train_bow_title)).tocsr()

X_cv=hstack((categories_ohe_cv,subcategories_ohe_cv,state_ohe_cv,grade_ohe_cv,prefix_ohe_cv,price_
cv,quantity_cv,tnp_cv,cv_bow_essay,cv_bow_title)).tocsr()

X_te=hstack((categories_ohe_test,subcategories_ohe_test,state_ohe_test,grade_ohe_test,prefix_ohe_te
st,price_test,quantity_test,tnp_test,test_bow_essay,test_bow_title)).tocsr()
```

In [92]:

```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

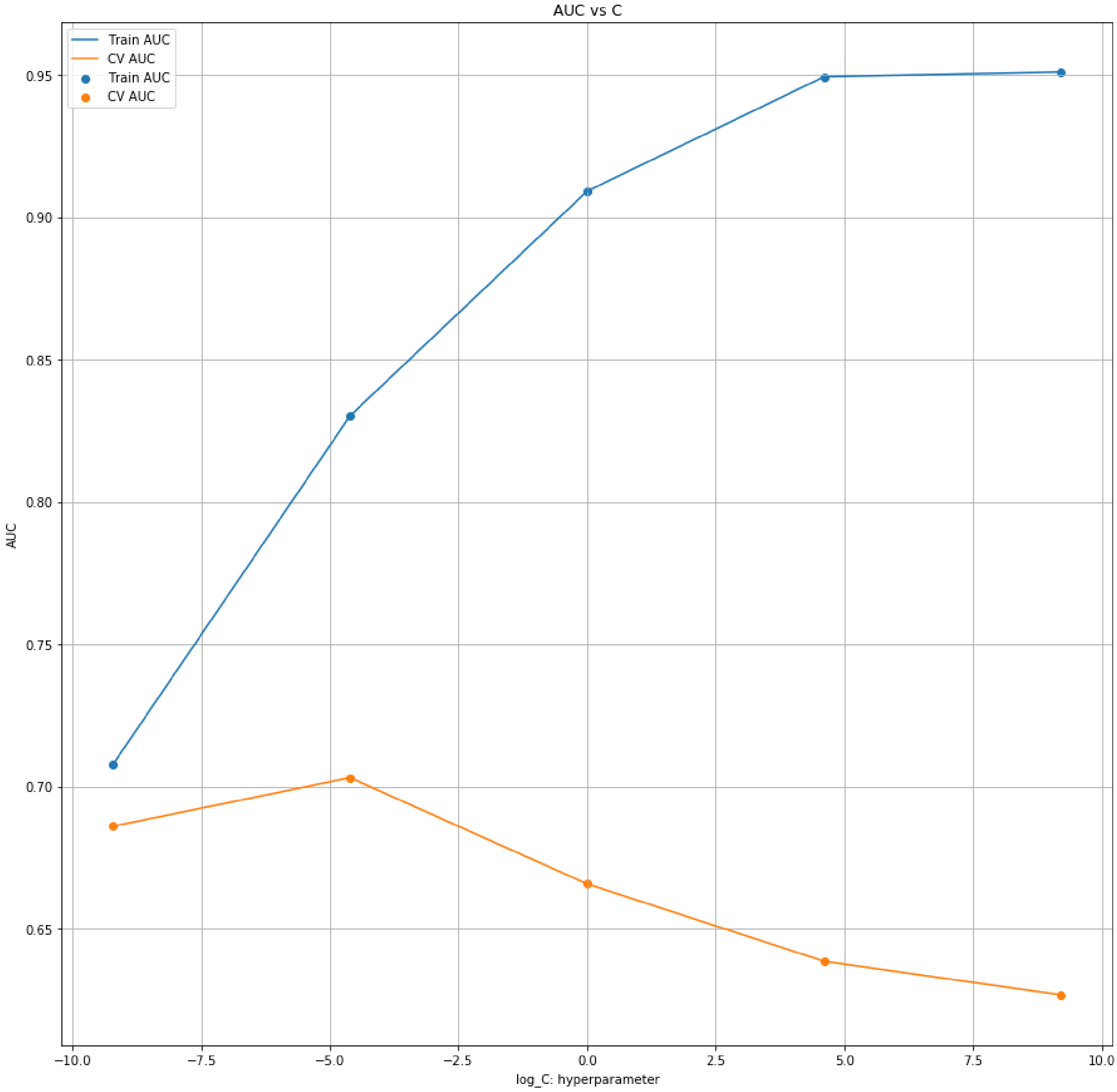
```
(49041, 16796) (49041,)
==================================================
(24155, 16796) (24155,)
==================================================
(36052, 16796) (36052,)
```

Finding best hyperparameter C

In [93]:

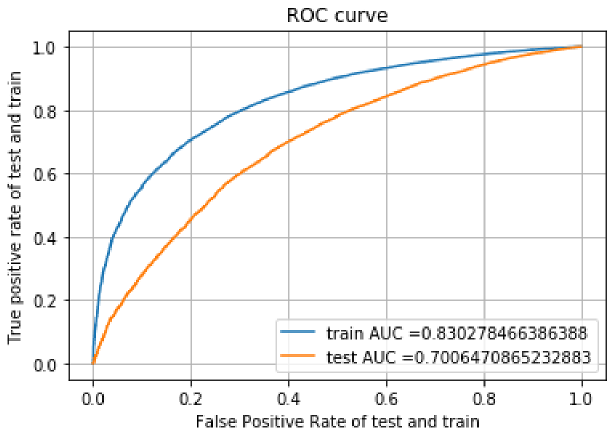
```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm
import math
alpha1=[]
train_auc = []
cv_auc = []
C = [10**-4, 10**-2, 10**0, 10**2, 10**4]
for i in tqdm(C):
    logreg = LogisticRegression(C=i, class_weight='balanced') #to deal with class imbalance
    logreg.fit(X_tr, y_train) #during fit our model is learning from the training data e.g. y=f(x)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    y_train_pred =  logreg.predict_log_proba(X_tr)[:,1]
    y_cv_pred =  logreg.predict_log_proba(X_cv)[:,1]
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pre
diction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

for i in C:
    j=math.log(i)
    alpha1.append(j)

plt.figure(figsize=(15,15))
plt.plot(alpha1, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(alpha1, train_auc, label='Train AUC') #Scatter plot of K vs auc train

plt.plot(alpha1, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(alpha1, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("log_C: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs C") #adding title of the plot
plt.grid()
plt.show()
```

```
100%|██████████████████████████████████████████████████████████████████████████████████| 5/5 [37:
55<00:00, 626.00s/it]
```



In [94]:

```
from sklearn.metrics import roc_curve,auc

logreg=LogisticRegression(C=10**-2,class_weight='balanced')
logreg.fit(X_tr,y_train)

#documentation of roc_curve ->https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc_curve returns three values fpr,tpr and thresholds
y_train_predict=logreg.predict_log_proba(X_tr)[:,1]
y_test_predict=logreg.predict_log_proba(X_te)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation ->https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("ROC curve")
plt.grid()
plt.show()
```



Confusion Matrix

we will be printing the confusion matrix for the threshold value which have low fpr and high tpr for this we will do the following steps :

1. Store the tpr fpr and threshold in a dataframe
2. create another columns to store specificity(1-fpr)
3. we will create another columns which will store the product of tpr and specificity
4. Sort the dataframe in descending order
5. with the help of binarize method we will calculate new probablitlities using that threshold which has maximum product of specificity and tpr

Train Data

In [95]:

```
df=pd.DataFrame({"fpr":train_fpr,"tpr":train_tpr,"threshold":train_thresholds})
print(df.head(3))
print(df.shape)
```

```
   fpr       tpr  threshold
0  0.0  0.000000   0.999975
1  0.0  0.000024  -0.000025
2  0.0  0.001442  -0.005905
(9982, 3)
```

In [96]:

```
df['Specificty']=1-df.fpr
```

In [97]:

```
df['Value']=df.tpr*df.Specificty
```

In [98]:

```
df.sort_values("Value", axis = 0, ascending = False,
                 inplace = True, na_position ='first')
df.head(3)
```

Out[98]:

	fpr	tpr	threshold	Specificty	Value
3145	0.232157	0.736441	-0.704006	0.767843	0.565471
3141	0.231888	0.736057	-0.703505	0.768112	0.565374
3295	0.243738	0.747519	-0.721212	0.756262	0.565320

In [99]:

```
index = df.Value.argmax()
a=df['threshold'][index]
print(a)
```

```
-0.7040058225310126
```

In [100]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_train_predict.reshape(-1,1),a)#changing the threshold and printing the first value
print(y_predict_thres[0])
```

```
[1.]
```

In [101]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("Train confusion matrix")
cm=confusion_matrix(y_train, y_predict_thres)
print(cm)
```

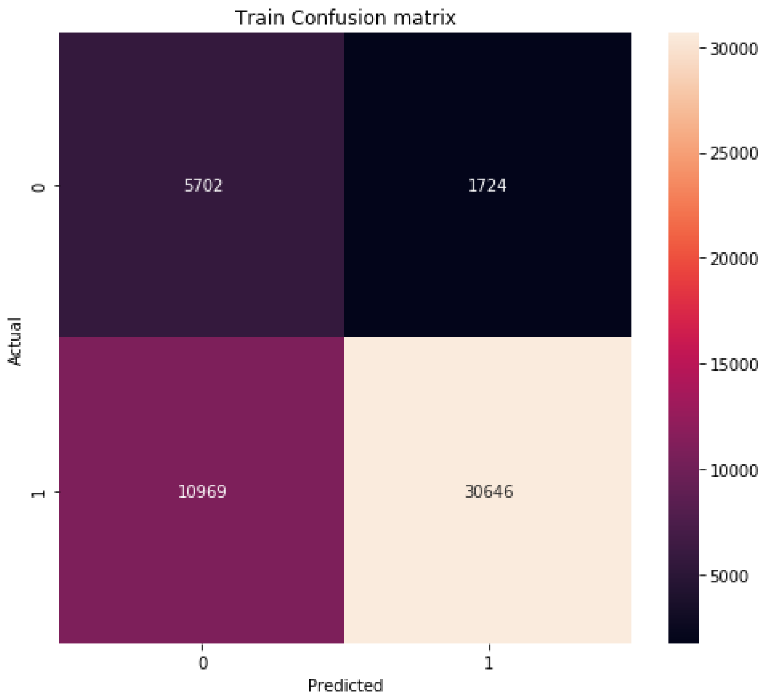
```
Threshold -0.7040058225310126
Train confusion matrix
[[ 5702  1724]
 [10969 30646]]
```

In [102]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[102]:

```
Text(0.5, 42.0, 'Predicted')
```



Test Data

In [103]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_test_predict.reshape(-1,1),a)#changing the threshold and printing the f
irst value
print(y_predict_thres[0])
```

```
[1.]
```

In [104]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)

print("Test confusion matrix")
cm1=confusion_matrix(y_test, y_predict_thres)
print(cm1)
```

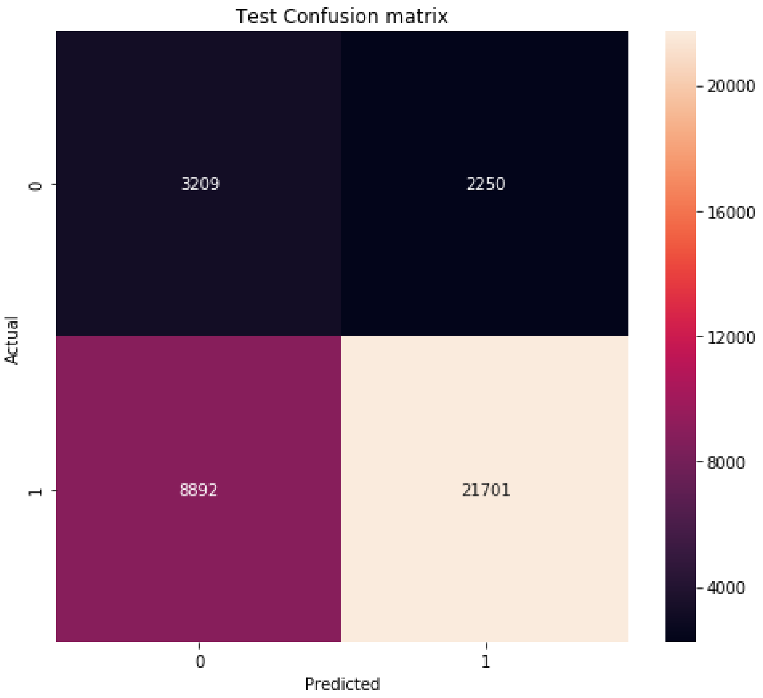
```
Threshold -0.7040058225310126
Test confusion matrix
[[ 3209  2250]
 [ 8892 21701]]
```

In [105]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[105]:

```
Text(0.5, 42.0, 'Predicted')
```



Set 2: Categorical Features,Numerical Features+Preprocessed Essay(tf-idf with unigram and bigram and max features 5000 and min df=10)+Preprocessed Title(tf-idf)

In [106]:

```
X_tr=hstack((categories_ohe_train,subcategories_ohe_train,state_ohe_train,grade_ohe_train,prefix_oh
e_train,price_train,quantity_train,tnp_train,train_tfidf_title,train_tfidf_essay)).tocsr()

X_cv=hstack((categories_ohe_cv,subcategories_ohe_cv,state_ohe_cv,grade_ohe_cv,prefix_ohe_cv,price_
cv,quantity_cv,tnp_cv,cv_tfidf_essay,cv_tfidf_title)).tocsr()

X_te=hstack((categories_ohe_test,subcategories_ohe_test,state_ohe_test,grade_ohe_test,prefix_ohe_te
st,price_test,quantity_test,tnp_test,test_tfidf_essay,test_tfidf_title)).tocsr()
```

In [107]:

```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

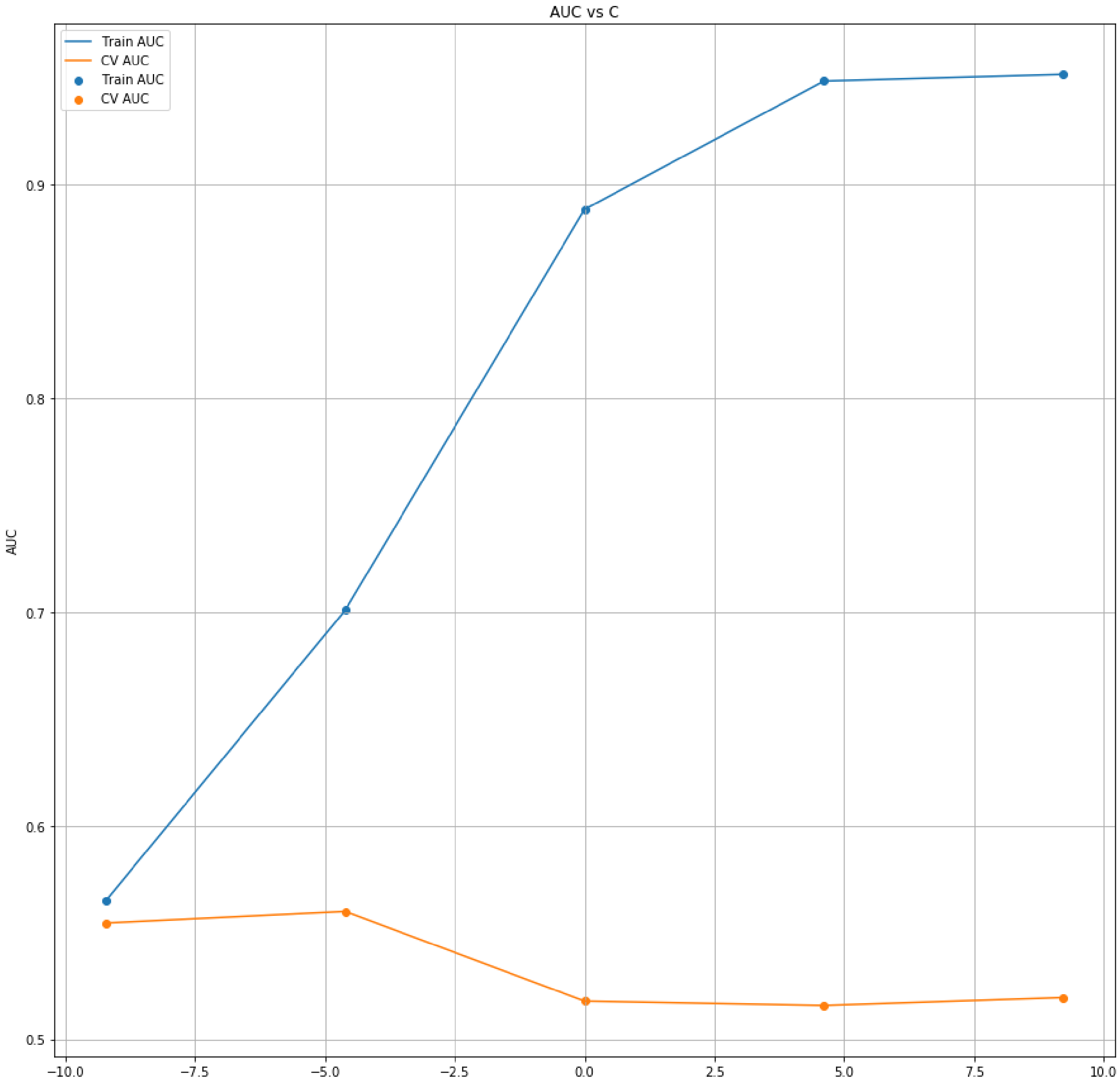
```
(49041, 16796) (49041,)
==================================================
(24155, 16796) (24155,)
==================================================
(36052, 16796) (36052,)
```

In [108]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm
alpha1=[]
train_auc = []
cv_auc = []
C = [10**-4, 10**-2, 10**0, 10**2, 10**4]
for i in tqdm(C):
    logreg = LogisticRegression(C=i, class_weight='balanced') #to deal with class imbalance
    logreg.fit(X_tr, y_train) #during fit our model is learning from the training data e.g. y=f(x)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
```

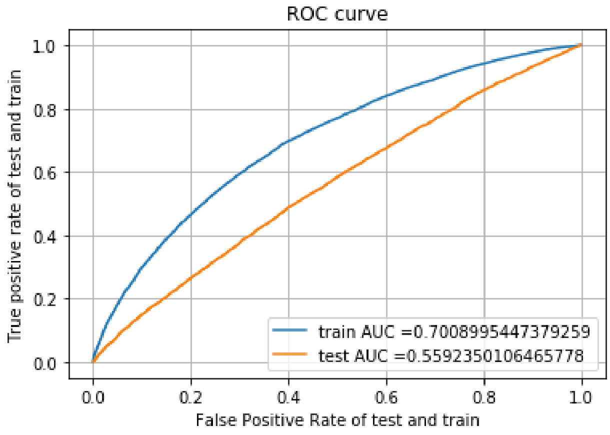
```
tive class
    # not the predicted outputs
    y_train_pred =  logreg.predict_log_proba(X_tr)[:,1]
    y_cv_pred =  logreg.predict_log_proba(X_cv)[:,1]
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pre
diction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

for i in C:
    j=math.log(i)
    alpha1.append(j)

plt.figure(figsize=(15,15))
plt.plot(alpha1, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(alpha1, train_auc, label='Train AUC') #Scatter plot of K vs auc train

plt.plot(alpha1, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(alpha1, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("log_C: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs C") #adding title of the plot
plt.grid()
plt.show()
```

```
100%|████████████████████████████████████████████████████████████████████████| 5/5 [13:
22<00:00, 224.76s/it]
```



In [109]:

```
from sklearn.metrics import roc_curve,auc

logreg=LogisticRegression(C=10**-2,class_weight='balanced')
logreg.fit(X_tr,y_train)

#documentation of roc_curve ->https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc_curve returns three values fpr,tpr and thresholds
y_train_predict=logreg.predict_log_proba(X_tr)[:,1]
y_test_predict=logreg.predict_log_proba(X_te)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation ->https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("ROC curve")
plt.grid()
plt.show()
```



Confusion Matrix

we will be printing the confusion matrix for the threshold value which have low fpr and high tpr for this we will do the following steps :

1. Store the tpr fpr and threshold in a dataframe
2. create another columns to store specificity(1-fpr)
3. we will create another columns which will store the product of tpr and specificity
4. Sort the dataframe in descending order
5. with the help of binarize method we will calculate new probablitlities using that threshold which has maximum product of specificity and tpr

Train Data

In [110]:

```
df=pd.DataFrame({"fpr":train_fpr,"tpr":train_tpr,"threshold":train_thresholds})
print(df.head(3))
print(df.shape)
```

```
   fpr       tpr  threshold
0  0.0  0.000000   0.718973
1  0.0  0.000024  -0.281027
```

```
2  0.0  0.003725  -0.370609
(11750, 3)
```

In [111]:

```
df['Specificty']=1-df.fpr
```

In [112]:

```
df['Value']=df.tpr*df.Specificty
```

In [113]:

```
df.sort_values("Value", axis = 0, ascending = False,
                 inplace = True, na_position ='first')
df.head(3)
```

Out[113]:

	fpr	tpr	threshold	Specificty	Value
5140	0.387961	0.687565	-0.710541	0.612039	0.420816
5088	0.383921	0.682903	-0.708675	0.616079	0.420722
5138	0.387827	0.687252	-0.710419	0.612173	0.420718

In [114]:

```
index = df.Value.argmax()
a=df['threshold'][index]
print(a)
```

```
-0.710541202619818
```

In [115]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_train_predict.reshape(-1,1),a)#changing the threshold and printing the
first value
print(y_predict_thres[0])
```

```
[1.]
```

In [116]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("Train confusion matrix")
cm=confusion_matrix(y_train, y_predict_thres)
print(cm)
```

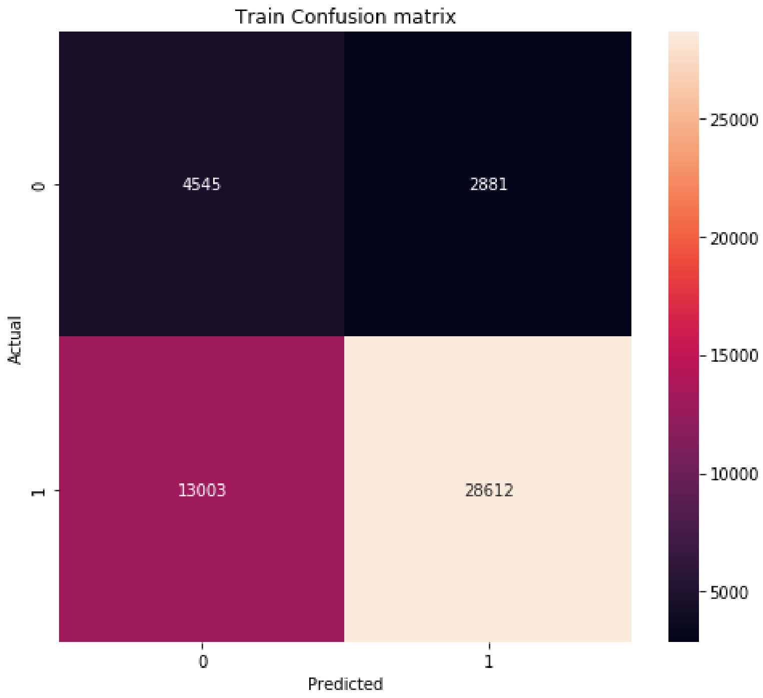
```
Threshold -0.710541202619818
Train confusion matrix
[[ 4545  2881]
 [13003 28612]]
```

In [117]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[117]:

```
Text(0.5, 42.0, 'Predicted')
```



Test Data

In [118]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_test_predict.reshape(-1,1),a)#changing the threshold and printing the f
irst value
print(y_predict_thres[0])
```

```
[0.]
```

In [119]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)

print("Test confusion matrix")
cm1=confusion_matrix(y_test, y_predict_thres)
print(cm1)
```

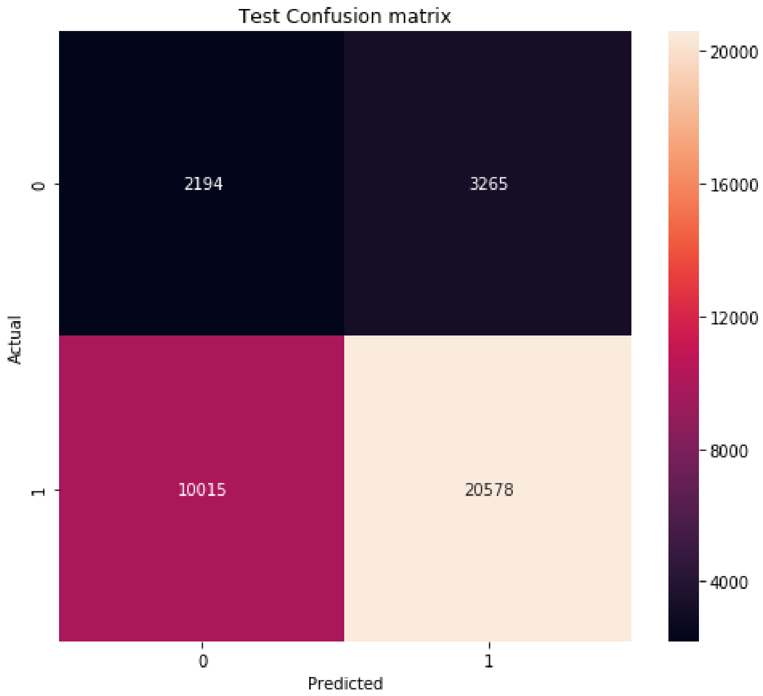
```
Threshold -0.710541202619818
Test confusion matrix
[[ 2194  3265]
 [10015 20578]]
```

In [120]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[120]:

```
Text(0.5, 42.0, 'Predicted')
```



Set 3: Categorical Features,Numerical Features+Preprocessed Essay(Avg W2V)+Preprocessed Title(Avg W2V)

In [121]:

```
X_tr=hstack((categories_ohe_train,subcategories_ohe_train,state_ohe_train,grade_ohe_train,prefix_ohe_train,price_train,quantity_train,tnp_train,avg_w2v_vectors_train,avg_w2v_vectors_title_train)).tocsr()

X_cv=hstack((categories_ohe_cv,subcategories_ohe_cv,state_ohe_cv,grade_ohe_cv,prefix_ohe_cv,price_cv,quantity_cv,tnp_cv,avg_w2v_vectors_cv,avg_w2v_vectors_title_cv)).tocsr()

X_te=hstack((categories_ohe_test,subcategories_ohe_test,state_ohe_test,grade_ohe_test,prefix_ohe_test,price_test,quantity_test,tnp_test,avg_w2v_vectors_test,avg_w2v_vectors_title_test)).tocsr()
```

In [122]:

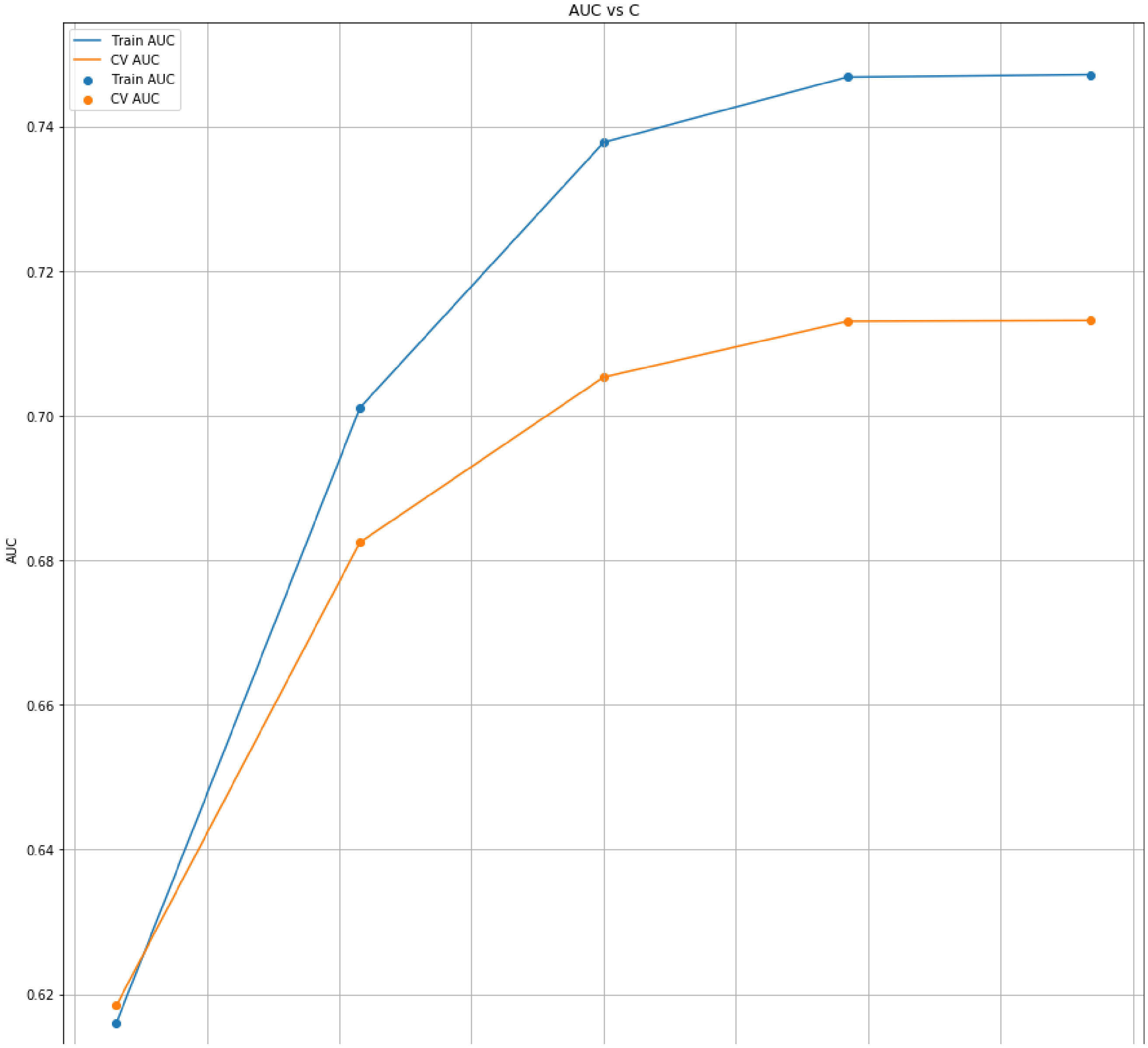
```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

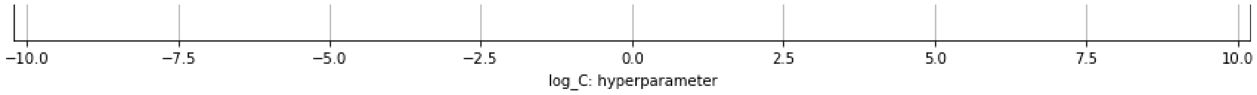
```
(49041, 703) (49041,)
==================================================
(24155, 703) (24155,)
==================================================
(36052, 703) (36052,)
```

In [123]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm
alpha1=[]
train_auc = []
cv_auc = []
C = [10**-4, 10**-2, 10**0, 10**2, 10**4]
for i in tqdm(C):
    logreg = LogisticRegression(C=i, class_weight='balanced') #to deal with class imbalance
    logreg = LogisticRegression(C=i, class_weight='balanced') #to deal with class imbalance
    logreg.fit(X_tr, y_train) #during fit our model is learning from the training data e.g. y=f(x)
    # 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 =  logreg.predict_log_proba(X_tr)[:,1]
    y_cv_pred =  logreg.predict_log_proba(X_cv)[:,1]
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from prediction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

for i in C:
    j=math.log(i)
    alpha1.append(j)

plt.figure(figsize=(15,15))
plt.plot(alpha1, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(alpha1, train_auc, label='Train AUC') #Scatter plot of K vs auc train

plt.plot(alpha1, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(alpha1, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("log_C: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs C") #adding title of the plot
plt.grid()
plt.show()
```

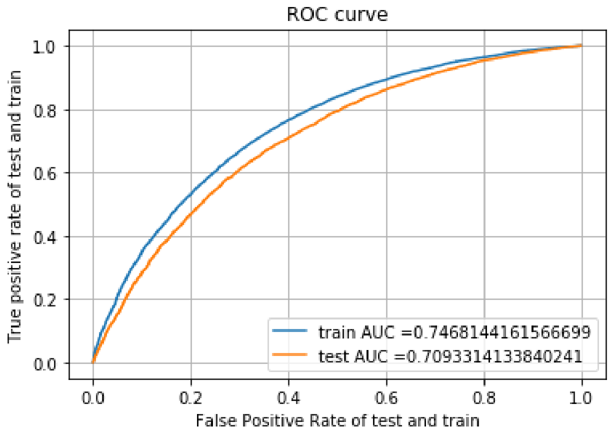
```
100%|████████████████████████████████████████████████████████████████████████████████| 5/5 [13:10<00:00, 191.25s/it]
```





In [124]:

```
from sklearn.metrics import roc_curve,auc

logreg=LogisticRegression(C=10**2,class_weight='balanced')
logreg.fit(X_tr,y_train)

#documentation of roc_curve ->https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc_curve returns three values fpr,tpr and thresholds
y_train_predict=logreg.predict_log_proba(X_tr)[:,1]
y_test_predict=logreg.predict_log_proba(X_te)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation ->https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("ROC curve")
plt.grid()
plt.show()
```



Confusion Matrix

we will be printing the confusion matrix for the threshold value which have low fpr and high tpr for this we will do the following steps :

1. Store the tpr fpr and threshold in a dataframe
2. create another columns to store specificity(1-fpr)
3. we will create another columns which will store the product of tpr and specificity
4. Sort the dataframe in descending order
5. with the help of binarize method we will calculate new probablitlities using that threshold which has maximum product of specificity and tpr

Train Data

In [125]:

```
df=pd.DataFrame({"fpr":train_fpr,"tpr":train_tpr,"threshold":train_thresholds})
print(df.head(3))
print(df.shape)
```

```
   fpr        tpr  threshold
   fpr       tpr  threshold
0  0.0  0.000000   0.997172
1  0.0  0.000024  -0.002828
2  0.0  0.000288  -0.007742
(11163, 3)
```

In [126]:

```
df['Specificty']=1-df.fpr
```

In [127]:

```
df['Value']=df.tpr*df.Specificty
```

In [128]:

```
df.sort_values("Value", axis = 0, ascending = False,
                 inplace = True, na_position ='first')
df.head(3)
```

Out[128]:

	fpr	tpr	threshold	Specificty	Value
4376	0.324401	0.694557	-0.718691	0.675599	0.469242
4378	0.324535	0.694629	-0.718777	0.675465	0.469198
4380	0.324670	0.694725	-0.718863	0.675330	0.469169

In [129]:

```
index = df.Value.argmax()
a=df['threshold'][index]
print(a)
```

```
-0.7186914342132779
```

In [130]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_train_predict.reshape(-1,1),a)#changing the threshold and printing the first value
print(y_predict_thres[0])
```

```
[1.]
```

In [131]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("Train confusion matrix")
cm=confusion_matrix(y_train, y_predict_thres)
print(cm)
```

```
Threshold -0.7186914342132779
Train confusion matrix
[[ 5017  2409]
 [12712 28903]]
```

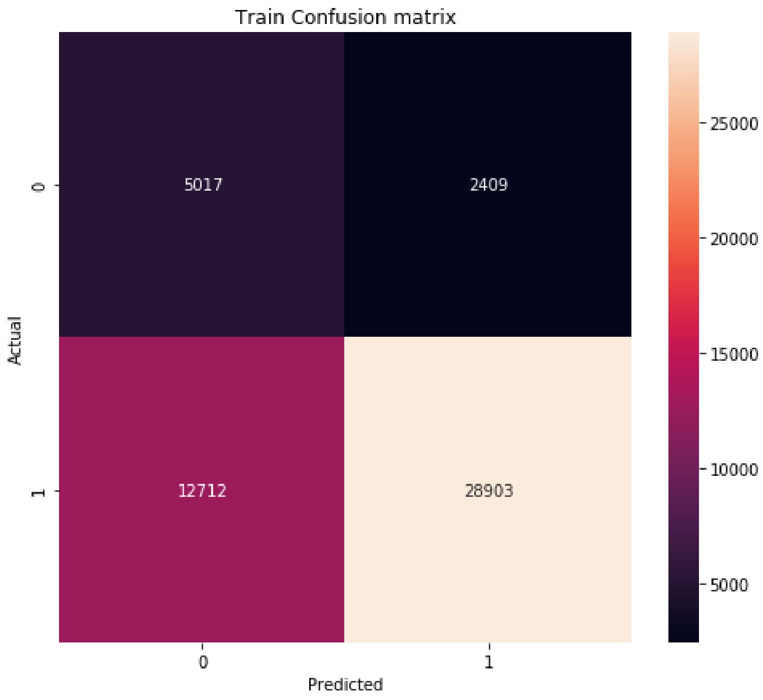
In [132]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
```

```
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[132]:

```
Text(0.5, 42.0, 'Predicted')
```



Test Data

In [133]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_test_predict.reshape(-1,1),a)#changing the threshold and printing the first value
print(y_predict_thres[0])
```

```
[1.]
```

In [134]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)

print("Test confusion matrix")
cm1=confusion_matrix(y_test, y_predict_thres)
print(cm1)
```

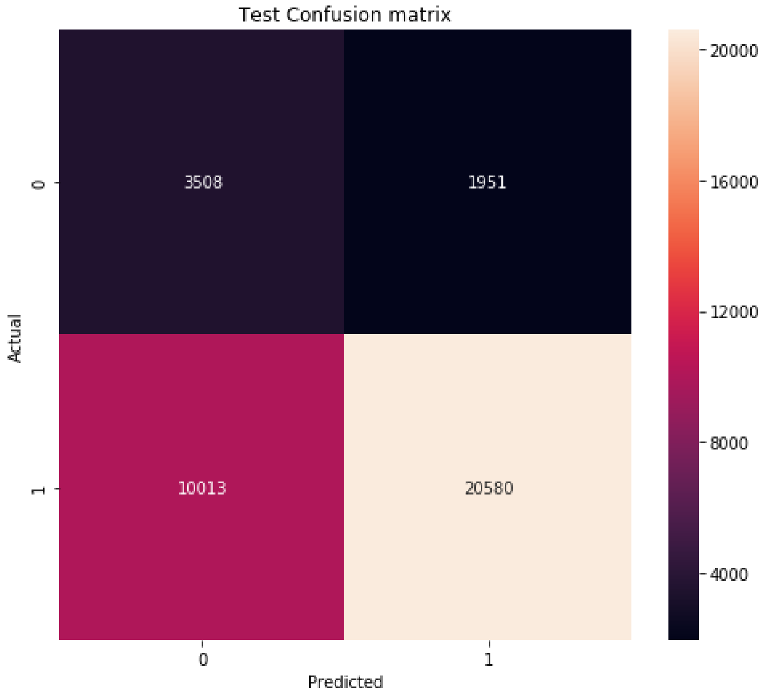
```
Threshold -0.7186914342132779
Test confusion matrix
[[ 3508  1951]
 [10013 20580]]
```

In [135]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[135]:

```
Text(0.5, 42.0, 'Predicted')
```



Set 4: Categorical Features,Numerical Features+Preprocessed Essay(TFIDF-W2V)+Preprocessed Title(TFIDF- W2V)

In [136]:

```
X_tr=hstack((categories_ohe_train,subcategories_ohe_train,state_ohe_train,grade_ohe_train,prefix_oh
e_train,price_train,quantity_train,tnp_train,tfidf_w2v_vectors_train,tfidf_w2v_vectors_title_train
)).tocsr()

X_cv=hstack((categories_ohe_cv,subcategories_ohe_cv,state_ohe_cv,grade_ohe_cv,prefix_ohe_cv,price_
cv,quantity_cv,tnp_cv,tfidf_w2v_vectors_cv,tfidf_w2v_vectors_title_cv)).tocsr()

X_te=hstack((categories_ohe_test,subcategories_ohe_test,state_ohe_test,grade_ohe_test,prefix_ohe_te
st,price_test,quantity_test,tnp_test,tfidf_w2v_vectors_test,tfidf_w2v_vectors_title_test)).tocsr()
```

In [137]:

```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

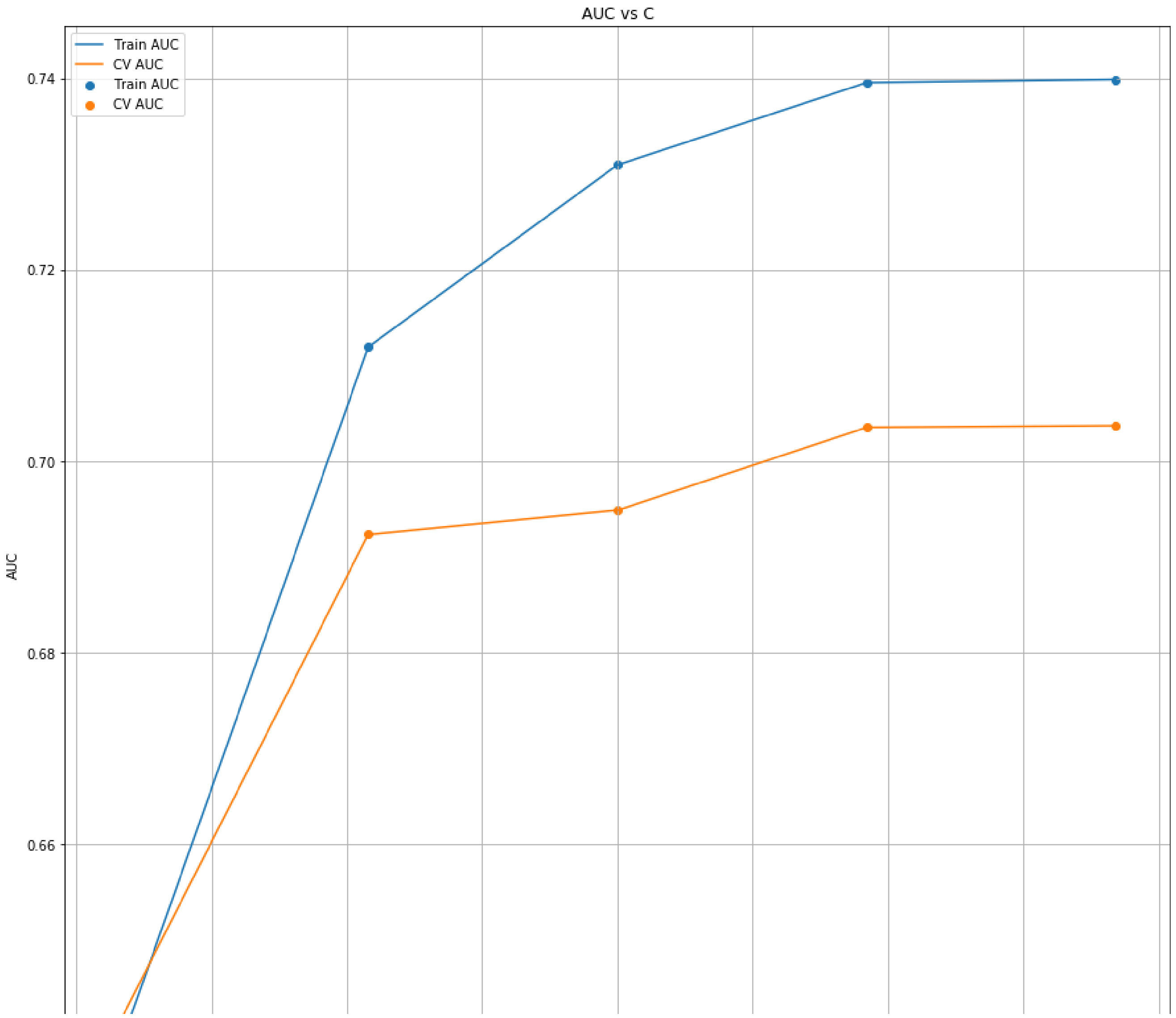
```
(49041, 703) (49041,)
==================================================
(24155, 703) (24155,)
==================================================
(36052, 703) (36052,)
```

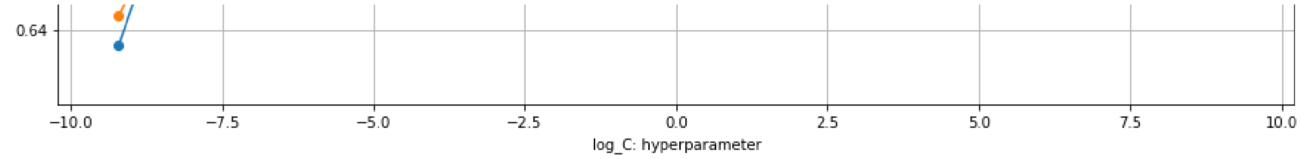
In [138]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm
alpha1=[]
train_auc = []
cv_auc = []
```

```
C = [10**-4, 10**-2, 10**0, 10**2, 10**4]
for i in tqdm(C):
    logreg = LogisticRegression(C=i, class_weight='balanced') #to deal with class imbalance
    logreg.fit(X_tr, y_train) #during fit our model is learning from the training data e.g. y=f(x)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    y_train_pred =  logreg.predict_log_proba(X_tr)[:,1]
    y_cv_pred =  logreg.predict_log_proba(X_cv)[:,1]
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pre
diction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

for i in C:
    j=math.log(i)
    alpha1.append(j)

plt.figure(figsize=(15,15))
plt.plot(alpha1, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(alpha1, train_auc, label='Train AUC') #Scatter plot of K vs auc train

plt.plot(alpha1, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(alpha1, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("log_C: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs C") #adding title of the plot
plt.grid()
plt.show()
```

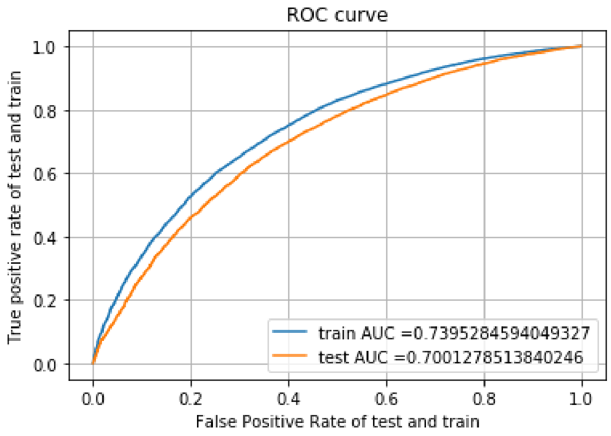
```
100%|██████████████████████████████████████████████████████████████████████████| 5/5 [13:
45<00:00, 207.30s/it]
```





In [139]:

```
from sklearn.metrics import roc_curve,auc

logreg=LogisticRegression(C=10**2,class_weight='balanced')
logreg.fit(X_tr,y_train)

#documentation of roc_curve ->https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc_curve returns three values fpr,tpr and thresholds
y_train_predict=logreg.predict_log_proba(X_tr)[:,1]
y_test_predict=logreg.predict_log_proba(X_te)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation ->https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("ROC curve")
plt.grid()
plt.show()
```



Confusion Matrix

we will be printing the confusion matrix for the threshold value which have low fpr and high tpr for this we will do the following steps :

1. Store the tpr fpr and threshold in a dataframe
2. create another columns to store specificity(1-fpr)
3. we will create another columns which will store the product of tpr and specificity
4. Sort the dataframe in descending order
5. with the help of binarize method we will calculate new probablitlities using that threshold which has maximum product of specificity and tpr

Train Data

In [140]:

```
df=pd.DataFrame({"fpr":train_fpr,"tpr":train_tpr,"threshold":train_thresholds})
print(df.head(3))
print(df.shape)
```

```
   fpr       tpr  threshold
0  0.0  0.000000   0.999092
1  0.0  0.000024  -0.000908
2  0.0  0.001466  -0.007647
(11320, 3)
```

In [141]:

```
df['Specificty']=1-df.fpr
```

In [142]:

```
df['Value']=df.tpr*df.Specificty
```

In [143]:

```
df.sort_values("Value", axis = 0, ascending = False,
                 inplace = True, na_position ='first')
df.head(3)
```

Out[143]:

	fpr	tpr	threshold	Specificty	Value
4264	0.314840	0.669206	-0.707801	0.685160	0.458513
4268	0.315244	0.669590	-0.708110	0.684756	0.458506
4262	0.314705	0.669062	-0.707591	0.685295	0.458505

In [144]:

```
index = df.Value.argmax()
a=df['threshold'][index]
print(a)
```

```
-0.7078013998773182
```

In [145]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_train_predict.reshape(-1,1),a)#changing the threshold and printing the
first value
print(y_predict_thres[0])
```

```
[1.]
```

In [146]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("Train confusion matrix")
cm=confusion_matrix(y_train, y_predict_thres)
print(cm)
```

```
Threshold -0.7078013998773182
Train confusion matrix
[[ 5088  2338]
 [13767 27848]]
```

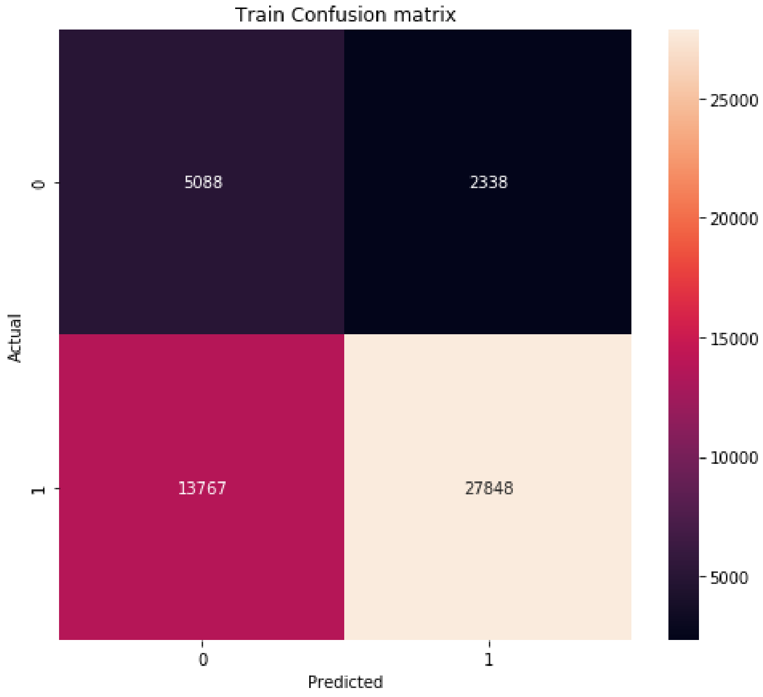
In [147]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
```

```
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[147]:

```
Text(0.5, 42.0, 'Predicted')
```



Test Data

In [148]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_test_predict.reshape(-1,1),a)#changing the threshold and printing the f
irst value
print(y_predict_thres[0])
```

```
[1.]
```

In [149]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)

print("Test confusion matrix")
cm1=confusion_matrix(y_test, y_predict_thres)
print(cm1)
```

```
Threshold -0.7078013998773182
Test confusion matrix
[[ 3574  1885]
 [10803 19790]]
```

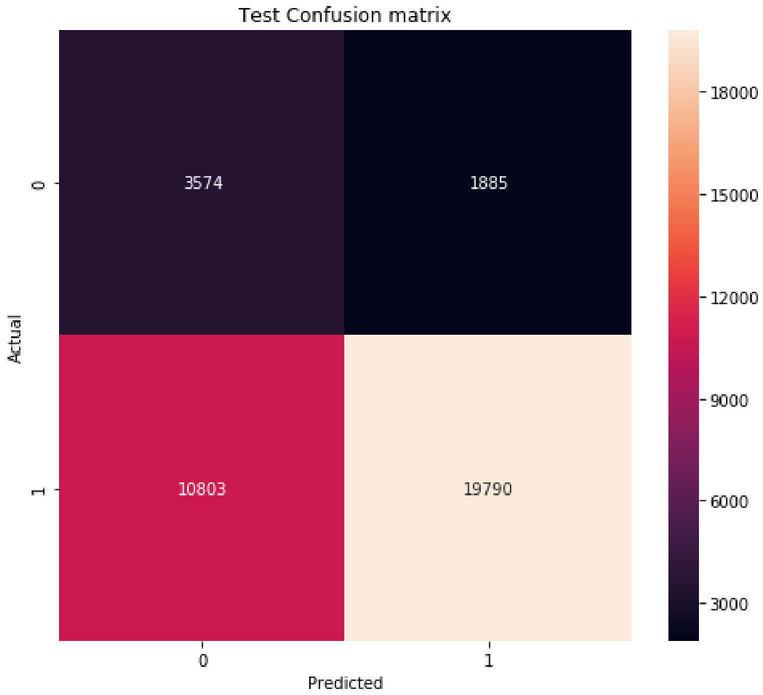
In [150]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
```

```
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[150]:

```
Text(0.5, 42.0, 'Predicted')
```



Set 5: Categorical Features + Numerical Features

In [151]:

```
X_tr=hstack((categories_ohe_train,subcategories_ohe_train,state_ohe_train,grade_ohe_train,prefix_oh
e_train,price_train,quantity_train,tnp_train)).tocsr()

X_cv=hstack((categories_ohe_cv,subcategories_ohe_cv,state_ohe_cv,grade_ohe_cv,prefix_ohe_cv,price_
cv,quantity_cv,tnp_cv)).tocsr()

X_te=hstack((categories_ohe_test,subcategories_ohe_test,state_ohe_test,grade_ohe_test,prefix_ohe_te
st,price_test,quantity_test,tnp_test)).tocsr()
```

In [152]:

```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

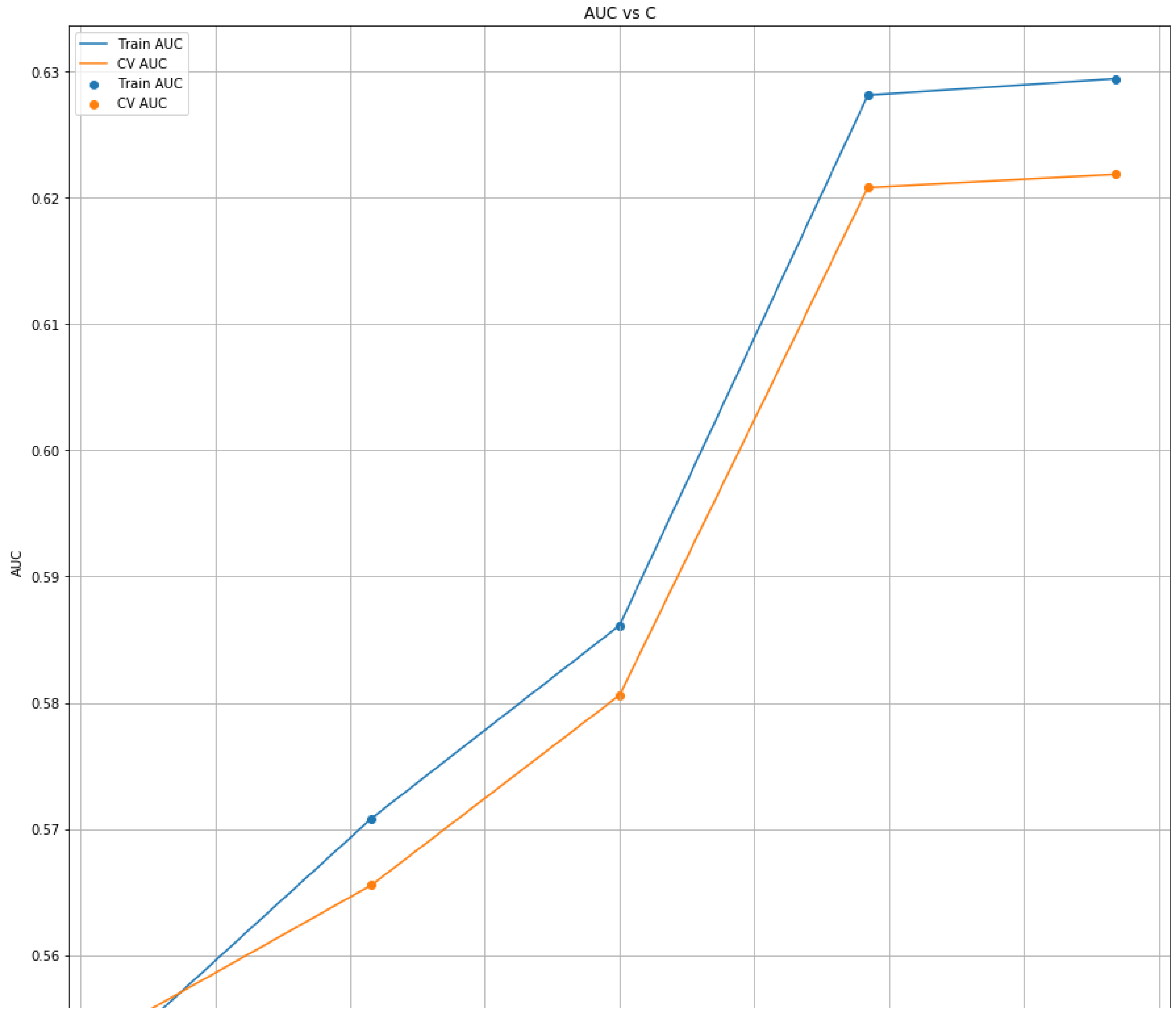
```
(49041, 103) (49041,)
==================================================
(24155, 103) (24155,)
==================================================
(36052, 103) (36052,)
```

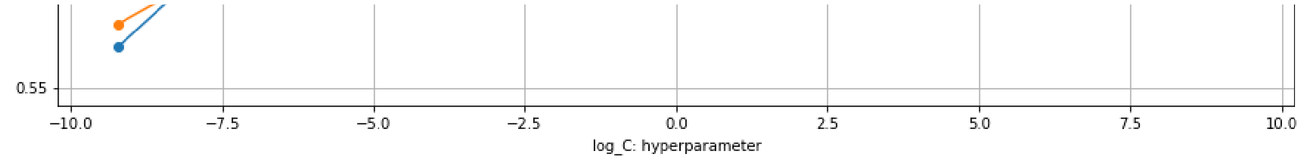
In [153]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm
alpha1=[]
train_auc = []
cv_auc = []
```

```
C = [10**-4, 10**-2, 10**0, 10**2, 10**4]
for i in tqdm(C):
    logreg = LogisticRegression(C=i, class_weight='balanced') #to deal with class imbalance
    logreg.fit(X_tr, y_train) #during fit our model is learning from the training data e.g. y=f(x)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    y_train_pred =  logreg.predict_log_proba(X_tr)[:,1]
    y_cv_pred =  logreg.predict_log_proba(X_cv)[:,1]
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pre
diction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

for i in C:
    j=math.log(i)
    alpha1.append(j)

plt.figure(figsize=(15,15))
plt.plot(alpha1, train_auc, label='Train AUC') #Plotting K vs auc of train
plt.scatter(alpha1, train_auc, label='Train AUC') #Scatter plot of K vs auc train

plt.plot(alpha1, cv_auc, label='CV AUC') #Plotting K vs auc of train
plt.scatter(alpha1, cv_auc, label='CV AUC') #Scatter plot of K vs auc train

plt.legend() #adding legend
plt.xlabel("log_C: hyperparameter") #X axis-label
plt.ylabel("AUC")  #Y-axis label
plt.title("AUC vs C") #adding title of the plot
plt.grid()
plt.show()
```

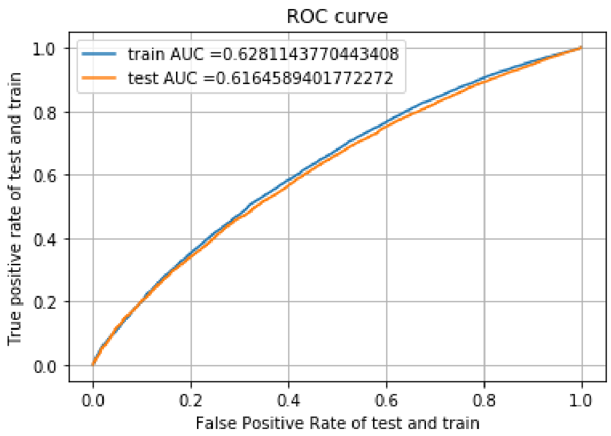
```
100%|██████████████████████████████████████████████████████████████████| 5/5 [00
:17<00:00,  4.10s/it]
```





In [154]:

```
from sklearn.metrics import roc_curve,auc

logreg=LogisticRegression(C=10**2,class_weight='balanced')
logreg.fit(X_tr,y_train)

#documentation of roc_curve ->https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
#roc_curve returns three values fpr,tpr and thresholds
y_train_predict=logreg.predict_log_proba(X_tr)[:,1]
y_test_predict=logreg.predict_log_proba(X_te)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,y_train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,y_test_predict)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr))) #documentation of auc-> https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.xlabel("False Positive Rate of test and train") #plt.plot documentation ->https://matplotlib.org/3.1.0/tutorials/introductory/pyplot.html
plt.ylabel("True positive rate of test and train")
plt.title("ROC curve")
plt.grid()
plt.show()
```



Confusion Matrix

we will be printing the confusion matrix for the threshold value which have low fpr and high tpr for this we will do the following steps :

1. Store the tpr fpr and threshold in a dataframe
2. create another columns to store specificity(1-fpr)
3. we will create another columns which will store the product of tpr and specificity
4. Sort the dataframe in descending order
5. with the help of binarize method we will calculate new probablitlities using that threshold which has maximum product of specificity and tpr

Train Data

In [155]:

```
df=pd.DataFrame({"fpr":train_fpr,"tpr":train_tpr,"threshold":train_thresholds})
print(df.head(3))
print(df.shape)
```

```
   fpr       tpr  threshold
0  0.0  0.000000   0.992309
1  0.0  0.000024  -0.007691
2  0.0  0.001346  -0.048749
(12437, 3)
```

In [156]:

```
df['Specificty']=1-df.fpr
```

In [157]:

```
df['Value']=df.tpr*df.Specificty
```

In [158]:

```
df.sort_values("Value", axis = 0, ascending = False,
                 inplace = True, na_position ='first')
df.head(3)
```

Out[158]:

	fpr	tpr	threshold	Specificty	Value
5315	0.396176	0.580416	-0.681039	0.603824	0.350469
5321	0.396849	0.580968	-0.681311	0.603151	0.350412
5311	0.395906	0.579983	-0.680882	0.604094	0.350364

In [159]:

```
index = df.Value.argmax()
a=df['threshold'][index]
print(a)
```

```
-0.6810389530322157
```

In [160]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_train_predict.reshape(-1,1),a)#changing the threshold and printing the
first value
print(y_predict_thres[0])
```

```
[1.]
```

In [161]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("Train confusion matrix")
cm=confusion_matrix(y_train, y_predict_thres)
print(cm)
```

```
Threshold -0.6810389530322157
Train confusion matrix
[[ 4484  2942]
 [17462 24153]]
```

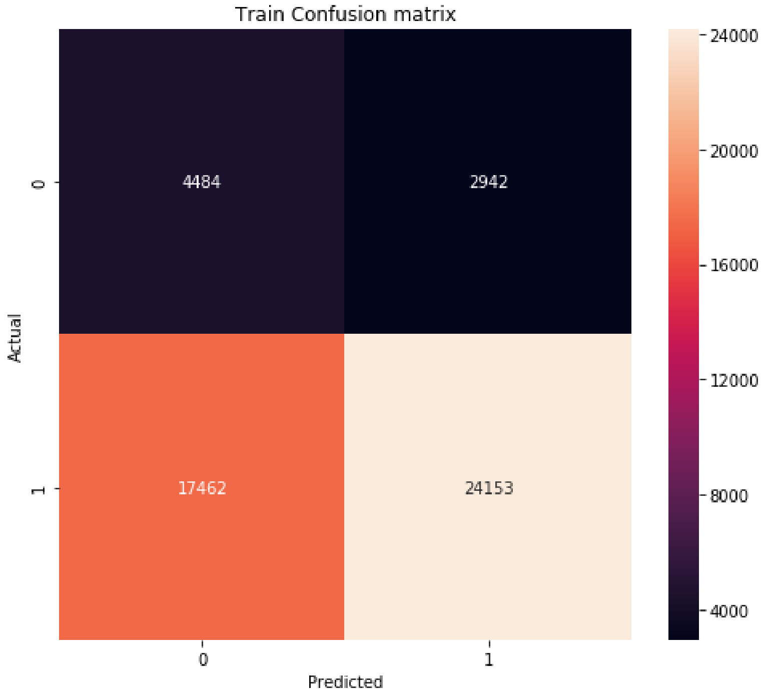
In [162]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm,index=[0,1],columns=[0,1])
```

```
plt.figure(figsize = (8,7))
plt.title("Train Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[162]:

```
Text(0.5, 42.0, 'Predicted')
```



Test Data

In [163]:

```
from sklearn.preprocessing import binarize
y_predict_thres=binarize(y_test_predict.reshape(-1,1),a)#changing the threshold and printing the f
irst value
print(y_predict_thres[0])
```

```
[0.]
```

In [164]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)

print("Test confusion matrix")
cm1=confusion_matrix(y_test, y_predict_thres)
print(cm1)
```

```
Threshold -0.6810389530322157
Test confusion matrix
[[ 3445  2014]
 [14307 16286]]
```

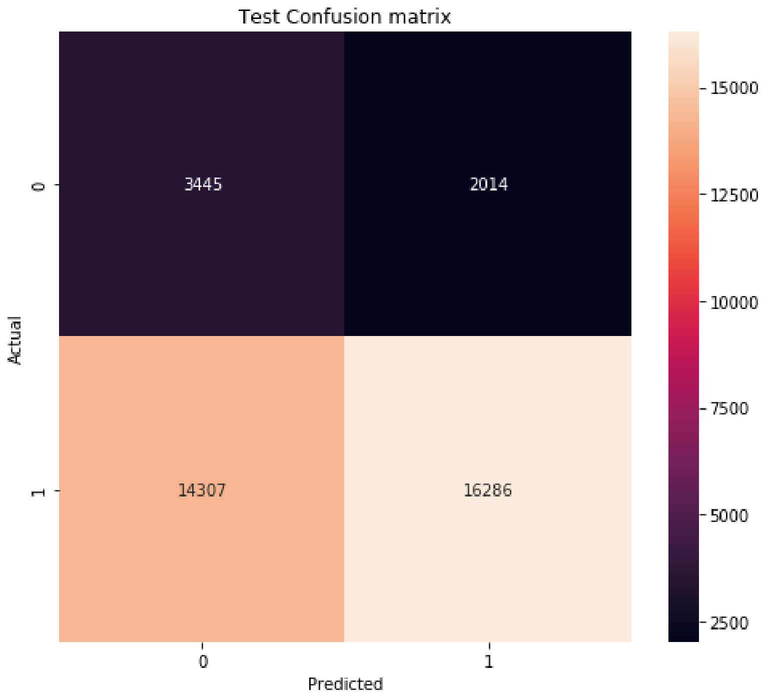
In [165]:

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix

import seaborn as sn
df_cm=pd.DataFrame(cm1,index=[0,1],columns=[0,1])
plt.figure(figsize = (8,7))
plt.title("Test Confusion matrix")
ax=sn.heatmap(df_cm, annot=True,fmt='g')
```

```
ax.set_ylabel("Actual")
ax.set_xlabel("Predicted")
```

Out[165]:

```
Text(0.5, 42.0, 'Predicted')
```



Summary

In [166]:

```
#Refer->http://zetcode.com/python/prettytable/
#Refer->https://het.as.utexas.edu/HET/Software/Numpy/reference/generated/numpy.percentile.html
#Refer->https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.round_.html
from prettytable import PrettyTable
x=PrettyTable()

x.field_names=["SET","Vectorizer", "Model", "Best Hyperparameter(C)","Test AUC"] #column headers

x.add_row(["I", "BOW",  "Brute", 10**-2 , 0.700])
x.add_row(["II", "TFIDF", "Brute",  10**-2 , 0.559])
x.add_row(["III", "AVG W2V", "Brute" ,10**2, 0.709])
x.add_row(["IV", "TFIDF" , "Brute", 10**2, 0.700])
x.add_row(["V", "Without-Text", "Brute", 10**2, 0.616])

print(x)
```

```
+-----+--------------+-------+------------------------+----------+
| SET |  Vectorizer  | Model | Best Hyperparameter(C) | Test AUC |
+-----+--------------+-------+------------------------+----------+
|  I  |     BOW      | Brute |          0.01          |   0.7    |
|  II |    TFIDF     | Brute |          0.01          |  0.559   |
| III |   AVG W2V    | Brute |          100           |  0.709   |
|  IV |    TFIDF     | Brute |          100           |   0.7    |
|  V  | Without-Text | Brute |          100           |  0.616   |
+-----+--------------+-------+------------------------+----------+
```

Without the inclusion of text documents in our training model the performance of the model decrease for both the train and testing stage as we can see on comparing the AUC scores

In [ ]: