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]:

```
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`
	An explanation of the resources needed for the project.**Example:** • `My students need hands on literacy materials to manage sensory`

Feature	Description
`project_resource_summary`	• `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*
`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.

Reading the data

In [3]:

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

In [4]:

```
print("datapoints in train data=",train_data.shape)
```

```
datapoints in train data= (109248, 17)
```

In [5]:

```
print("column names",train_data.columns)
```

```
column names 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')
```

In [6]:

```python
print(train_data.head())
```

```
   Unnamed: 0       id                        teacher_id teacher_prefix  \
0      160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
1      140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.
2       21895  p182444  3465aaf82da834c0582ebd0ef8040ca0            Ms.
3          45  p246581  f3cb9bffbba169bef1a77b243e620b60           Mrs.
4      172407  p104768  be1f7507a41f8479dc06f047086a39ec           Mrs.

  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
3           KY        2016-10-06 21:16:17          Grades PreK-2
4           TX        2016-07-11 01:10:09          Grades PreK-2

            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
3  Literacy & Language, Math & Science             Literacy, Mathematics
4                       Math & Science                       Mathematics

                                       project_title  \
0   Educational Support for English Learners at Home
1              Wanted: Projector for Hungry Learners
2  Soccer Equipment for AWESOME Middle School Stu...
3                             Techie Kindergarteners
4                             Interactive Math Tools

                                     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...
3  I work at a unique school filled with both ESL...
4  Our second grade classroom next year will be m...

                                     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
3  My students live in high poverty conditions wi...             NaN
4  For many students, math is a subject that does...             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,...
3             NaN  My students need to engage in Reading and Math...
4             NaN  My students need hands on practice in mathemat...

   teacher_number_of_previously_posted_projects  project_is_approved
0                                             0                    0
1                                             7                    1
2                                             1                    0
3                                             4                    1
4                                             1                    1
```

In [7]:

```python
# 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 encounters 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[7]:

	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 [9]:

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

```
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   
3  p069063  Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...         2   
4  p069063  EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...         3   

    price  
0  149.00  
1   14.95  
2    8.45  
3   13.59  
4   24.95  
```

In [10]:

```
#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  
0         4  
```

In [11]:

```
#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
```

In [12]:

```
train_data=train_data.iloc[0:90000,:]
print(train_data.shape)
```

```
(90000, 19)
```

In [13]:

```
train_data1=train_data.iloc[0:30000,:]
print(train_data1.shape)
```

```
(30000, 19)
```

In [14]:

```
train_data1.columns
```

Out[14]:

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

Feature Preprocessing

Preprocessing of project_subject_categories

Train_Data

In [15]:

```
print(train_data.project_subject_categories[0:5])
print("*"*50)
categories=list(train_data["project_subject_categories"].values)#created a list of the values in t
he project_subject_categories
print(categories[0:5])
```

```
0         Math & Science
1          Special Needs
2    Literacy & Language
3       Applied Learning
4    Literacy & Language
Name: project_subject_categories, dtype: object
**************************************************
['Math & Science', 'Special Needs', 'Literacy & Language', 'Applied Learning', 'Literacy &
Language']
```

In [16]:

```
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[0:5])
```

```
['mathscience', 'specialneeds', 'literacylanguage', 'appliedlearning', 'literacylanguage']
```

In [17]:

```
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 [18]:

```
# 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
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': 33999, 'specialneeds': 11081, 'literacylanguage': 43236, 'appliedlearning': 9706,
'historycivics': 4821, 'musicarts': 8490, 'healthsports': 12693, 'warmth': 770, 'carehunger': 770}
==================================================
--------------------------------------------------
[('appliedlearning', 9706), ('carehunger', 770), ('healthsports', 12693), ('historycivics', 4821),
('literacylanguage', 43236), ('mathscience', 33999), ('musicarts', 8490), ('specialneeds', 11081),
('warmth', 770)]
```

Preprocessing of project_subject_subcategories

In [19]:

```
print(train_data.project_subject_subcategories[0:5])
print("*"*50)
categories=list(train_data["project_subject_subcategories"].values)#created a list of the values i
n the project_subject_categories
print(categories[0:5])
```

```
0    Applied Sciences, Health & Life Science
1                              Special Needs
2                                   Literacy
3                          Early Development
4                                   Literacy
Name: project_subject_subcategories, dtype: object
**************************************************
['Applied Sciences, Health & Life Science', 'Special Needs', 'Literacy', 'Early Development', 'Lit
eracy']
```

In [20]:

```
#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[0:5])
```

```
['appliedsciences healthlifescience', 'specialneeds', 'literacy', 'earlydevelopment', 'literacy']
```

In [21]:

```
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 [22]:

```
# 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/
```

```
# 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': 8754, 'healthlifescience': 3473, 'specialneeds': 11081, 'literacy': 27763,
'earlydevelopment': 3344, 'mathematics': 23150, 'socialsciences': 1622, 'historygeography': 2658,
'esl': 3590, 'extracurricular': 623, 'visualarts': 5220, 'environmentalscience': 4550, 'literaturew
riting': 18522, 'gymfitness': 4029, 'music': 2588, 'teamsports': 1717, 'performingarts': 1567,
'collegecareerprep': 2120, 'other': 1903, 'charactereducation': 1705, 'foreignlanguages': 730, 'he
althwellness': 9398, 'civicsgovernment': 693, 'economics': 220, 'communityservice': 372,
'financialliteracy': 325, 'nutritioneducation': 1296, 'parentinvolvement': 491, 'warmth': 770, 'ca
rehunger': 770}
==================================================
[('appliedsciences', 8754), ('carehunger', 770), ('charactereducation', 1705),
('civicsgovernment', 693), ('collegecareerprep', 2120), ('communityservice', 372),
('earlydevelopment', 3344), ('economics', 220), ('environmentalscience', 4550), ('esl', 3590), ('e
xtracurricular', 623), ('financialliteracy', 325), ('foreignlanguages', 730), ('gymfitness',
4029), ('healthlifescience', 3473), ('healthwellness', 9398), ('historygeography', 2658),
('literacy', 27763), ('literaturewriting', 18522), ('mathematics', 23150), ('music', 2588),
('nutritioneducation', 1296), ('other', 1903), ('parentinvolvement', 491), ('performingarts', 1567
), ('socialsciences', 1622), ('specialneeds', 11081), ('teamsports', 1717), ('visualarts', 5220),
('warmth', 770)]
```

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

Train_Data

In [23]:

```
# 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 [24]:

```
# printing some random essays.
print(train_data['project_essay'].values[10])
print("="*50)
print(train_data['project_essay'].values[20000])
print("="*50)
print(train_data['project_essay'].values[942])
print("="*50)
print(train_data['project_essay'].values[451])
print("="*50)
print(train_data['project_essay'].values[99])
print("="*50)
```

```
My students yearn for a classroom environment that matches their desire to learn. With education c
hanging daily, we need a classroom that can meet the needs of all of my first graders.I have the p
rivilege of teaching an incredible group of six and seven year olds who absolutely LOVE to learn.
I am completely blown away by their love for learning. Each day is a new adventure as they enjoy l
earning from nonfiction text and hands on activities. Many of my students are very active learners
who benefit from kinesthetic activities. Sometimes learning, while sitting in a seat, is
difficult. I want every child the opportunity to focus their energy in order to do their best in
school!Ideally, I would love to delve right into \"flexible seating\" where students are provided
```

```
many different seating options (chairs, hokki stools, on mats on the ground, etc.) and they have t
he freedom to choose which ever seat they feel they need. My student would be able to choose which
seating option will best help them learn. In addition, a pencil sharpener, mobile easel, magnetic
strips and mounting tape will help make our classroom better suited for 6 and 7 year olds.This pro
ject will be so beneficial for my students in that they will be able to better focus their energy.
Something so small, choosing their own seat, will help encourage a positive learning environment t
hat promotes learning for all students. The easel will help make our classroom more mobile, becaus
e it is both dry erase and on wheels. Magnetic strips, mounting tape and a pencil sharpener will a
llow for more resources for the students during the school day.
==================================================
\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the bi
ggest enthusiasm for learning. My students learn in many different ways using all of our senses an
d multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nSt
udents in my class come from a variety of different backgrounds which makes for wonderful sharing
of experiences and cultures, including Native Americans.\r\nOur school is a caring community of su
ccessful learners which can be seen through collaborative student project based learning in and ou
t of the classroom. Kindergarteners in my class love to work with hands-on materials and have many
different opportunities to practice a skill before it is mastered. Having the social skills to wor
k cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love to role play in our
pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try coo
king with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it's healthy for their bodies. This project w
ould expand our learning of nutrition and agricultural cooking recipes by having us peel our own a
pples to make homemade applesauce, make our own bread, and mix up healthy plants from our classroo
m garden in the spring. We will also create our own cookbooks to be printed and shared with famili
es. \r\nStudents will gain math and literature skills as well as a life long enjoyment for healthy
cooking.nannan
==================================================
Can you imagine sitting still for hours on end? I can't do that as an adult and I certainly don't
expect my students to be able to either!I teach at a school with a very diverse population. We hav
e students from every many ethnicity and backgrounds. Our school is between 2 major cities. Many s
tudents receive free or reduced lunches and we have a good size military population. \r\nI love my
class but they are very bouncy and love to move!I want to offer my students the choice to sit in t
he seats they want! They currently sit in hard plastic chairs that are NOT comfortable! I want the
m to be comfortable and be able to wiggle around and use energy, which promotes brain power! Each
morning they will have the chance to pick their seat so they can start the day of right!This proje
ct will make a difference because research has shown that the more kids move - the more they learn
! By giving them as many opportunities as possible toe move (even when in their seats) I can help
them live up to their full potential!
==================================================
\"If kids come to us from strong, healthy functioning families, it makes our job easier. If they d
o not come to us from strong, healthy, functioning families, it makes our job more
important.\"~Barbara Colorose.My students are housed in a Life Skills Unit, which is considered th
e most restricted due to their behaviors and/or disabilities.  We are a public high school located
in a high-poverty area. We are avid participants in Special Olympics and Community Based
Instruction.Many students at our school come hungry and our resources are limited. I would be able
to provide a healthy snack to those in need. I would also use as positive motivators throughout
the day. I would use many of the snacks as counting items in order to engage my students with extr
a needs.  The trail mix is great for sorting, classifying and graphing.This project will improve m
y classroom because I cannot always afford to buy the snacks I would like to have as motivators.
Sometimes, a little snack is all that is needed to get them back on track and ready to learn.
==================================================
A typical lesson in my school starts with a read aloud from a picture book to introduce the
reading or writing tasks students are learning.  These read-alouds serve as mentors in the
learning process.Units of study in Reading and Writing are the curricular guides at my project-bas
ed, Reggio-inspired elementary school.  Students are eager to learn a new teaching point each day,
which is usually inspired by the context of the daily read-aloud.  The texts allow us to talk
about our shared reading experience, since the students love to chatter!When the students have acc
ess to quality read-alouds that strongly relate to our daily teaching point, they are able to expe
rience the academic standard in the realistic context of literature.  For example, literacy expert
Katie Wood Ray advises using the book Beekeepers as an example that exhibits what writers do when
they share a slice of their life.  These books and guides offer unlimited lessons about what good
readers and writers do.Your donation will allow students to live in the worlds of these books!  Th
ey will be able to participate in memorable lessons that engage their minds.  Read-alouds can be t
he key to hooking them into learning about reading and writing.
==================================================
```

In [25]:

```
# 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 [26]:

```
test = decontracted(train_data['project_essay'].values[20000])
print(test)
print("="*50)
```

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the b
iggest enthusiasm for learning. My students learn in many different ways using all of our senses a
nd multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nS
tudents in my class come from a variety of different backgrounds which makes for wonderful sharing
of experiences and cultures, including Native Americans.\r\nOur school is a caring community of su
ccessful learners which can be seen through collaborative student project based learning in and ou
t of the classroom. Kindergarteners in my class love to work with hands-on materials and have many
different opportunities to practice a skill before it is mastered. Having the social skills to wor
k cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love to role play in our
pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try coo
king with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes by having us peel our own
apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classro
om garden in the spring. We will also create our own cookbooks to be printed and shared with famil
ies. \r\nStudents will gain math and literature skills as well as a life long enjoyment for health
y cooking.nannan
==================================================
```

In [27]:

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

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the smallest students with the big
gest enthusiasm for learning. My students learn in many different ways using all of our senses and
multiple intelligences. I use a wide range of techniques to help all my students succeed.
Students in my class come from a variety of different backgrounds which makes for wonderful
sharing of experiences and cultures, including Native Americans.  Our school is a caring community
of successful learners which can be seen through collaborative student project based learning in a
nd out of the classroom. Kindergarteners in my class love to work with hands-on materials and have
many different opportunities to practice a skill before it is mastered. Having the social skills t
o work cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is
the perfect place to learn about agriculture and nutrition. My students love to role play in our p
retend kitchen in the early childhood classroom. I have had several kids ask me,  Can we try cooki
ng with REAL food?  I will take their idea and create  Common Core Cooking Lessons  where we learn
important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes by having us peel our own
apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classro
om garden in the spring. We will also create our own cookbooks to be printed and shared with famil
ies.   Students will gain math and literature skills as well as a life long enjoyment for healthy
cooking.nannan
```

In [28]:

In [28]:

```
#remove special character: https://stackoverflow.com/a/5843547/4084039
test = re.sub('[^A-Za-z0-9]+', ' ', test) #square bracket creates either or set; + signifes 1 or m
ore character
print(test)
```

```
 A person is a person no matter how small Dr Seuss I teach the smallest students with the biggest
enthusiasm for learning My students learn in many different ways using all of our senses and multi
ple intelligences I use a wide range of techniques to help all my students succeed Students in my
class come from a variety of different backgrounds which makes for wonderful sharing of
experiences and cultures including Native Americans Our school is a caring community of successful
learners which can be seen through collaborative student project based learning in and out of the
classroom Kindergarteners in my class love to work with hands on materials and have many different
opportunities to practice a skill before it is mastered Having the social skills to work
cooperatively with friends is a crucial aspect of the kindergarten curriculum Montana is the
perfect place to learn about agriculture and nutrition My students love to role play in our
pretend kitchen in the early childhood classroom I have had several kids ask me Can we try cooking
with REAL food I will take their idea and create Common Core Cooking Lessons where we learn
important math and writing concepts while cooking delicious healthy food for snack time My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies This project w
ould expand our learning of nutrition and agricultural cooking recipes by having us peel our own a
pples to make homemade applesauce make our own bread and mix up healthy plants from our classroom
garden in the spring We will also create our own cookbooks to be printed and shared with families
Students will gain math and literature skills as well as a life long enjoyment for healthy cooking
nannan
```

In [29]:

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

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

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

In [31]:

```
#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%|██████████████████████████████████████████████████████████████████████████| 90000/90000
[00:14<00:00, 6273.09it/s]
```

In [32]:

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

Out[32]:

```
'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 [33]:

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

Project title text

In [34]:

```
# Printing some random project title
# printing some random essays.
print(train_data['project_title'].values[7])
print("="*50)
print(train_data['project_title'].values[9])
print("="*50)
print(train_data['project_title'].values[16])
print("="*50)
print(train_data['project_title'].values[23])
print("="*50)
```

```
21st Century Learning with Multimedia
==================================================
Dash and Dot Robotic Duo Needed
==================================================
Help us travel the world...VIRTUALLY!
==================================================
Techies in Training
==================================================
```

In [35]:

```
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%|█████████████████████████████████████████████████████████████████████████| 90000/90000
[00:01<00:00, 61432.39it/s]
```

In [36]:

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

Category Preprocessing

Train_Data

Teacher Prefix

In [37]:

```
train_data['teacher_prefix'].head(5) #printing the first 5 values to see what preprocessing should be made
```

Out[37]:

```
0     Mrs.
1      Ms.
2     Mrs.
3     Mrs.
4     Mrs.
Name: teacher_prefix, dtype: object
```

In [38]:

```
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%|██████████████████████████████████████████████████████████████████████| 90000/90000
[00:00<00:00, 1428653.35it/s]
```

In [39]:

```
preprocessed_prefix[3]
```

Out[39]:

```
'mrs'
```

In [40]:

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

Grade Category

In [41]:

```
train_data['project_grade_category'].head(5) #printing the first 5 values to see what
preprocessing should be made
```

Out[41]:

```
0     Grades PreK-2
1        Grades 3-5
2     Grades PreK-2
3     Grades PreK-2
```

```
3     Grades PreK-2
4       Grades 3-5
Name: project_grade_category, dtype: object
```

In [42]:

```
train_data['project_grade_category'].value_counts()
```

Out[42]:

```
Grades PreK-2    36239
Grades 3-5       30835
Grades 6-8       14005
Grades 9-12       8921
Name: project_grade_category, dtype: int64
```

In [43]:

```
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%|██████████████████████████████████████████████████████████████████████| 90000/90000
[00:00<00:00, 1267020.97it/s]
```

In [44]:

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

Out[44]:

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

In [45]:

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

project_resource_summary

In [46]:

```
train_data['project_resource_summary'].head(5)
```

Out[46]:

```
0    My students need STEM kits to learn critical s...
1    My students need Boogie Boards for quiet senso...
2    My students need a mobile listening center to ...
3    My students need flexible seating in the class...
4    My students need copies of the New York Times ...
Name: project_resource_summary, dtype: object
```

In [47]:

```
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%|██████████████████████████████████████████████████████████████████████████| 90000/90000
[00:02<00:00, 40276.24it/s]
```

In [48]:

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

Out[48]:

```
['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 [49]:

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

Train_Data

In [50]:

```
train_data.columns
```

Out[50]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state', 'Date',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'teacher_number_of_previously_posted_projects',
       'project_is_approved', 'price', 'quantity', 'clean_categories',
       'clean_subcategories', 'preprocessed_essays', 'preprocessed_title',
       'preprocessed_prefix', 'preprocessed_grade', 'preprocessed_resource'],
      dtype='object')
```

In [51]:

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

In [52]:

```
print(X.columns)
print("*"*50)
print(X.head())
```

```
Index(['Unnamed: 0', 'school_state',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'preprocessed_essays', 'preprocessed_title', 'preprocessed_prefix',
       'preprocessed_grade', 'preprocessed_resource'],
      dtype='object')
**************************************************
   Unnamed: 0 school_state  teacher_number_of_previously_posted_projects  \
0        8393           CA                                            53
1       37728           UT                                             4
2       74477           CA                                            10
3      100660           GA                                             2
4       33679           WA                                             2

   project_is_approved   price  quantity  clean_categories  \
0                    1  725.05         4       mathscience
```

```
1                    1  213.03         8      specialneeds
2                    1  329.00         1  literacylanguage
3                    1  481.04         9   appliedlearning
4                    1   17.74        14  literacylanguage

                 clean_subcategories  \
0  appliedsciences healthlifescience   
1                       specialneeds   
2                           literacy   
3                   earlydevelopment   
4                           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...   
3  recently read article giving students choice l...   
4  students crave challenge eat obstacles breakfa...   

                        preprocessed_title preprocessed_prefix  \
0      engineering steam primary classroom                 mrs   
1                      sensory tools focus                  ms   
2  mobile learning mobile listening center                 mrs   
3       flexible seating flexible learning                 mrs   
4            going deep art inner thinking                 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...  
3      Grades_PreK_2  students need flexible seating classroom choos...  
4         Grades_3_5  students need copies new york times best selle...  
```

In [53]:

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

In [54]:

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

In [55]:

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

```
(90000, 12)
==================================================
(90000,)
```

In [56]:

```
X.columns
```

Out[56]:

```
Index(['Unnamed: 0', 'school_state',
       'teacher_number_of_previously_posted_projects', 'price', 'quantity',
       'clean_categories', 'clean_subcategories', 'preprocessed_essays',
       'preprocessed_title', 'preprocessed_prefix', 'preprocessed_grade',
       'preprocessed_resource'],
      dtype='object')
```

Data Splitting into train,cv and test

In [57]:

```
# ============================== 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.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 [58]:

```
# 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 [59]:

```
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 [60]:

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

print("="*100)
```

```
(40401, 12) (40401,)
(19899, 12) (19899,)
(29700, 12) (29700,)
====================================================================================================
```

Vectorization

Response Encoding of categorical feature

Category Feature

In [61]:

```
init_data=pd.DataFrame(columns=['categories','label'])
```

In [62]:

```
init_data['categories']=X_train['clean_categories']
init_data['label']=y_train
```

In [63]:

```
print(init_data.head())
print(init_data.shape)
```

```
                           categories  label
48062                     mathscience      1
81103      historycivics healthsports      1
19797  literacylanguage specialneeds      1
86367                literacylanguage      1
64405                     mathscience      1
(40401, 2)
```

In [64]:

```
#how to calculate conditional probability python pandas ->https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-in-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
```

In [65]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div(cond_prob , axis=0, level='categories'),columns=['probability'])
```

In [66]:

```
print(encoded_cat.head())
print(encoded_cat.tail())
```

```
                                        probability
label categories
0     appliedlearning                      0.171131
      appliedlearning healthsports         0.185345
      appliedlearning historycivics        0.212121
      appliedlearning literacylanguage     0.137019
      appliedlearning mathscience          0.161376
                                      probability
label categories
1     specialneeds                       0.804536
      specialneeds healthsports          0.823529
      specialneeds musicarts             0.789916
      specialneeds warmth carehunger     0.818182
      warmth carehunger                  0.902357
```

In [67]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[67]:

```
(98, 3)
```

In [68]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
```

In [69]:

```
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label                        categories  probability
0      0                   appliedlearning     0.171131
1      0      appliedlearning healthsports     0.185345
2      0     appliedlearning historycivics     0.212121
3      0  appliedlearning literacylanguage     0.137019
4      0       appliedlearning mathscience     0.161376
(48, 3)
    label                        categories  probability
48      1                   appliedlearning     0.828869
49      1      appliedlearning healthsports     0.814655
50      1     appliedlearning historycivics     0.787879
51      1  appliedlearning literacylanguage     0.862981
52      1       appliedlearning mathscience     0.838624
(50, 3)
```

In [70]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [71]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  0
idx= 0
idx = 1
temp1=  1
idx= 1
idx = 2
temp1=  2
idx= 2
idx = 3
temp1=  3
idx= 3
idx = 4
temp1=  4
idx= 4
idx = 5
temp1=  5
idx= 5
idx = 6
temp1=  6
idx= 6
idx = 7
temp1=  7
idx= 7
idx = 8
temp1=  8
idx= 8
idx = 9
temp1=  9
idx= 9
idx = 10
temp1=  10
idx= 10
idx = 11
temp1=  11
idx= 11
idx = 12
temp1=  12
idx= 12
idx = 13
temp1=  13
idx= 13
idx = 14
temp1=  14
idx= 14
```

```
idx= 14
idx = 15
temp1=  16
idx= 15
idx = 16
temp1=  17
idx= 16
idx = 17
temp1=  18
idx= 17
idx = 18
temp1=  19
idx= 18
idx = 19
temp1=  20
idx= 19
idx = 20
temp1=  21
idx= 20
idx = 21
temp1=  22
idx= 21
idx = 22
idx = 23
temp1=  23
idx= 23
idx = 24
temp1=  24
idx= 24
idx = 25
temp1=  25
idx= 25
idx = 26
temp1=  26
idx= 26
idx = 27
temp1=  27
idx= 27
idx = 28
temp1=  28
idx= 28
idx = 29
temp1=  29
idx= 29
idx = 30
temp1=  31
idx= 30
idx = 31
temp1=  32
idx= 31
idx = 32
temp1=  33
idx= 32
idx = 33
temp1=  34
idx= 33
idx = 34
temp1=  35
idx= 34
idx = 35
temp1=  36
idx= 35
idx = 36
temp1=  37
idx= 36
idx = 37
temp1=  38
idx= 37
idx = 38
temp1=  39
idx= 38
idx = 39
temp1=  40
idx= 39
idx = 40
temp1=  41
idx= 40
idx = 41
idx = 41
temp1=  42
idx= 41
idx = 42
temp1=  43
idx= 42
idx = 43
temp1=  45
idx= 43
idx = 44
temp1=  46
idx= 44
idx = 45
temp1=  47
idx= 45
idx = 46
temp1=  48
idx= 46
idx = 47
temp1=  49
idx= 47
idx = 48
idx = 49
```

In [73]:

```
c_0=[]
c_1=[]
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])

print(len(c_0))
print(len(c_1))
```

```
50
50
```

In [74]:

```
a=X_train['clean_categories'].unique()
a=a[0:50]
a.shape
```

Out[74]:

```
(50,)
```

In [75]:

```
#Creating A Response Table
res_table=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table['prob_0']=c_0
res_table['prob_1']=c_1
#res_table['label']=label
res_table['categories']=a
```

In [76]:

```
res_table.shape
```

Out[76]:

```
(50, 3)
```

Training based on response_table

Train Data

In [77]:

```
train_coded_cat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [78]:

```
temp_0=[]
temp_1=[]
for cat in X_train["clean_categories"].values:
    if cat in res_table["categories"].values:
        temp_0.append(res_table.loc[cat,"prob_0"])
        temp_1.append(res_table.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [79]:

```
train_coded_cat["prob_0"]=temp_0
train_coded_cat["prob_1"]=temp_1
```

In [81]:

```
train_coded_cat.shape
```

Out[81]:

```
(40401, 2)
```

CV Data

In [82]:

```
cv_coded_cat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [83]:

```
temp_0=[]
temp_1=[]
for cat in X_cv["clean_categories"].values:
    if cat in res_table["categories"].values:
        temp_0.append(res_table.loc[cat,"prob_0"])
        temp_1.append(res_table.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [84]:

```
cv_coded_cat["prob_0"]=temp_0
cv_coded_cat["prob_1"]=temp_1
```

In [85]:

```
cv_coded_cat.shape
```

Out[85]:

```
(19899, 2)
```

Test Data

In [86]:

```
test_coded_cat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [87]:

```
temp_0=[]
temp_1=[]
for cat in X_test["clean_categories"].values:
    if cat in res_table["categories"].values:
        temp_0.append(res_table.loc[cat,"prob_0"])
        temp_1.append(res_table.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [88]:

```
test_coded_cat["prob_0"]=temp_0
test_coded_cat["prob_1"]=temp_1
```

In [89]:

```
test_coded_cat.shape
```

Out[89]:

```
(29700, 2)
```

Sub_category

In [90]:

```
init_data=pd.DataFrame(columns=['categories','label'])
init_data['categories']=X_train['clean_subcategories']
init_data['label']=y_train
```

In [91]:

```
print(init_data.head())
print(init_data.shape)
```

```
                                categories  label
48062       appliedsciences mathematics      1
81103   civicsgovernment healthwellness      1
19797    literaturewriting specialneeds      1
86367                 literaturewriting      1
64405  environmentalscience mathematics      1
(40401, 2)
```

In [92]:

```
#how to calculate conditional probability python pandas -
>https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
n-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
```

In [93]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div
(cond_prob , axis=0, level='categories'),columns=['probability'])
```

In [94]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[94]:

```
(644, 3)
```

In [95]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
```

In [96]:

```
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label                          categories  probability
0      0                     appliedsciences     0.187160
1      0  appliedsciences charactereducation     0.157895
2      0    appliedsciences civicsgovernment     0.125000
3      0   appliedsciences collegecareerprep     0.167785
4      0    appliedsciences earlydevelopment     0.112903
(278, 3)
     label                          categories  probability
278      1                     appliedsciences     0.812840
279      1  appliedsciences charactereducation     0.842105
280      1    appliedsciences civicsgovernment     0.875000
281      1   appliedsciences collegecareerprep     0.832215
282      1    appliedsciences communityservice     1.000000
(366, 3)
```

In [97]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [98]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  0
idx= 0
idx = 1
temp1=  1
idx= 1
idx = 2
temp1=  2
idx= 2
idx = 3
temp1=  3
idx= 3
```

```
idx = 4
temp1=  5
idx= 4
idx = 5
temp1=  6
idx= 5
idx = 6
temp1=  7
idx= 6
idx = 7
temp1=  8
idx= 7
idx = 8
temp1=  9
idx= 8
idx = 9
temp1=  10
idx= 9
idx = 10
temp1=  11
idx= 10
idx = 11
temp1=  12
idx= 11
idx = 12
temp1=  13
idx= 12
idx = 13
temp1=  14
idx= 13
idx = 14
temp1=  15
idx= 14
idx = 15
temp1=  16
idx= 15
idx = 16
temp1=  17
idx= 16
idx = 17
temp1=  18
idx= 17
idx = 18
temp1=  19
idx= 18
idx = 19
temp1=  21
idx= 19
idx = 20
temp1=  22
idx= 20
idx = 21
temp1=  23
idx= 21
idx = 22
temp1=  24
idx= 22
idx = 23
temp1=  25
idx= 23
idx = 24
temp1=  26
idx= 24
idx = 25
temp1=  27
idx= 25
idx = 26
idx = 27
temp1=  28
idx= 27
idx = 28
temp1=  29
idx= 28
idx = 29
temp1=  30
idx= 29
idx = 30
```

```
temp1=  31
idx= 30
idx = 31
temp1=  32
idx= 31
idx = 32
idx = 33
temp1=  33
idx= 33
idx = 34
temp1=  34
idx= 34
idx = 35
temp1=  35
idx= 35
idx = 36
temp1=  36
idx= 36
idx = 37
temp1=  38
idx= 37
idx = 38
temp1=  39
idx= 38
idx = 39
temp1=  40
idx= 39
idx = 40
temp1=  41
idx= 40
idx = 41
temp1=  42
idx= 41
idx = 42
temp1=  43
idx= 42
idx = 43
temp1=  44
idx= 43
idx = 44
temp1=  46
idx= 44
idx = 45
temp1=  47
idx= 45
idx = 46
temp1=  49
idx= 46
idx = 47
temp1=  50
idx= 47
idx = 48
temp1=  51
idx= 48
idx = 49
temp1=  52
idx= 49
idx = 50
temp1=  53
idx= 50
idx = 51
temp1=  54
idx= 51
idx = 52
temp1=  56
idx= 52
idx = 53
temp1=  57
idx= 53
idx = 54
temp1=  58
idx= 54
idx = 55
temp1=  63
idx= 55
idx = 56
temp1=  65
```

```
idx= 56
idx = 57
temp1=  66
idx= 57
idx = 58
temp1=  67
idx= 58
idx = 59
temp1=  69
idx= 59
idx = 60
temp1=  70
idx= 60
idx = 61
temp1=  71
idx= 61
idx = 62
temp1=  74
idx= 62
idx = 63
temp1=  76
idx= 63
idx = 64
temp1=  78
idx= 64
idx = 65
temp1=  79
idx= 65
idx = 66
temp1=  80
idx= 66
idx = 67
temp1=  81
idx= 67
idx = 68
temp1=  82
idx= 68
idx = 69
temp1=  83
idx= 69
idx = 70
temp1=  84
idx= 70
idx = 71
temp1=  85
idx= 71
idx = 72
temp1=  86
idx= 72
idx = 73
temp1=  87
idx= 73
idx = 74
temp1=  88
idx= 74
idx = 75
temp1=  89
idx= 75
idx = 76
temp1=  90
idx= 76
idx = 77
temp1=  91
idx= 77
idx = 78
temp1=  92
idx= 78
idx = 79
temp1=  94
idx= 79
idx = 80
temp1=  95
idx= 80
idx = 81
temp1=  96
idx= 81
idx = 82
```

```
temp1=  97
idx= 82
idx = 83
temp1=  98
idx= 83
idx = 84
temp1=  101
idx= 84
idx = 85
temp1=  103
idx= 85
idx = 86
temp1=  105
idx= 86
idx = 87
temp1=  106
idx= 87
idx = 88
temp1=  107
idx= 88
idx = 89
temp1=  108
idx= 89
idx = 90
temp1=  109
idx= 90
idx = 91
idx = 92
temp1=  111
idx= 92
idx = 93
temp1=  115
idx= 93
idx = 94
temp1=  116
idx= 94
idx = 95
temp1=  117
idx= 95
idx = 96
temp1=  118
idx= 96
idx = 97
temp1=  119
idx= 97
idx = 98
idx = 99
temp1=  122
idx= 99
idx = 100
temp1=  123
idx= 100
idx = 101
temp1=  124
idx= 101
idx = 102
temp1=  126
idx= 102
idx = 103
temp1=  127
idx= 103
idx = 104
temp1=  128
idx= 104
idx = 105
temp1=  129
idx= 105
idx = 106
temp1=  130
idx= 106
idx = 107
temp1=  131
idx= 107
idx = 108
temp1=  132
idx= 108
idx = 109
```

```
temp1=  134
idx= 109
idx = 110
temp1=  135
idx= 110
idx = 111
temp1=  136
idx= 111
idx = 112
temp1=  140
idx= 112
idx = 113
temp1=  150
idx= 113
idx = 114
temp1=  151
idx= 114
idx = 115
temp1=  154
idx= 115
idx = 116
temp1=  155
idx= 116
idx = 117
temp1=  156
idx= 117
idx = 118
temp1=  157
idx= 118
idx = 119
temp1=  158
idx= 119
idx = 120
temp1=  159
idx= 120
idx = 121
temp1=  160
idx= 121
idx = 122
temp1=  161
idx= 122
idx = 123
temp1=  162
idx= 123
idx = 124
temp1=  163
idx= 124
idx = 125
temp1=  164
idx= 125
idx = 126
temp1=  165
idx= 126
idx = 127
temp1=  166
idx= 127
idx = 128
temp1=  167
idx= 128
idx = 129
temp1=  168
idx= 129
idx = 130
temp1=  169
idx= 130
idx = 131
temp1=  170
idx= 131
idx = 132
temp1=  172
idx= 132
idx = 133
temp1=  173
idx= 133
idx = 134
temp1=  175
idx= 134
```

```
idx = 135
temp1=  177
idx= 135
idx = 136
temp1=  178
idx= 136
idx = 137
temp1=  179
idx= 137
idx = 138
temp1=  180
idx= 138
idx = 139
temp1=  184
idx= 139
idx = 140
temp1=  185
idx= 140
idx = 141
temp1=  186
idx= 141
idx = 142
temp1=  187
idx= 142
idx = 143
temp1=  189
idx= 143
idx = 144
temp1=  190
idx= 144
idx = 145
idx = 146
temp1=  195
idx= 146
idx = 147
temp1=  196
idx= 147
idx = 148
temp1=  197
idx= 148
idx = 149
temp1=  198
idx= 149
idx = 150
temp1=  200
idx= 150
idx = 151
temp1=  201
idx= 151
idx = 152
temp1=  204
idx= 152
idx = 153
temp1=  205
idx= 153
idx = 154
temp1=  206
idx= 154
idx = 155
temp1=  207
idx= 155
idx = 156
temp1=  209
idx= 156
idx = 157
temp1=  212
idx= 157
idx = 158
temp1=  215
idx= 158
idx = 159
temp1=  217
idx= 159
idx = 160
temp1=  219
idx= 160
idx = 161
```

```
temp1=  221
idx= 161
idx = 162
temp1=  222
idx= 162
idx = 163
temp1=  223
idx= 163
idx = 164
temp1=  224
idx= 164
idx = 165
idx = 166
temp1=  228
idx= 166
idx = 167
temp1=  229
idx= 167
idx = 168
idx = 169
temp1=  230
idx= 169
idx = 170
temp1=  232
idx= 170
idx = 171
temp1=  233
idx= 171
idx = 172
temp1=  234
idx= 172
idx = 173
temp1=  236
idx= 173
idx = 174
temp1=  237
idx= 174
idx = 175
temp1=  239
idx= 175
idx = 176
temp1=  240
idx= 176
idx = 177
temp1=  241
idx= 177
idx = 178
temp1=  242
idx= 178
idx = 179
temp1=  243
idx= 179
idx = 180
temp1=  244
idx= 180
idx = 181
temp1=  245
idx= 181
idx = 182
temp1=  246
idx= 182
idx = 183
temp1=  247
idx= 183
idx = 184
temp1=  248
idx= 184
idx = 185
temp1=  250
idx= 185
idx = 186
temp1=  251
idx= 186
idx = 187
temp1=  252
idx= 187
idx = 188
```

```
temp1=  254
idx= 188
idx = 189
temp1=  255
idx= 189
idx = 190
temp1=  256
idx= 190
idx = 191
temp1=  257
idx= 191
idx = 192
temp1=  258
idx= 192
idx = 193
temp1=  259
idx= 193
idx = 194
temp1=  260
idx= 194
idx = 195
temp1=  261
idx= 195
idx = 196
temp1=  262
idx= 196
idx = 197
temp1=  263
idx= 197
idx = 198
temp1=  264
idx= 198
idx = 199
temp1=  265
idx= 199
idx = 200
temp1=  266
idx= 200
idx = 201
temp1=  267
idx= 201
idx = 202
temp1=  270
idx= 202
idx = 203
temp1=  271
idx= 203
idx = 204
temp1=  272
idx= 204
idx = 205
temp1=  274
idx= 205
idx = 206
temp1=  275
idx= 206
idx = 207
temp1=  276
idx= 207
idx = 208
temp1=  277
idx= 208
idx = 209
temp1=  279
idx= 209
idx = 210
temp1=  282
idx= 210
idx = 211
temp1=  283
idx= 211
idx = 212
temp1=  284
idx= 212
idx = 213
temp1=  285
idx= 213
```

```
idx = 214
idx = 215
temp1=  286
idx= 215
idx = 216
temp1=  287
idx= 216
idx = 217
temp1=  288
idx= 217
idx = 218
temp1=  289
idx= 218
idx = 219
temp1=  291
idx= 219
idx = 220
temp1=  292
idx= 220
idx = 221
temp1=  293
idx= 221
idx = 222
temp1=  294
idx= 222
idx = 223
temp1=  295
idx= 223
idx = 224
temp1=  297
idx= 224
idx = 225
temp1=  299
idx= 225
idx = 226
temp1=  300
idx= 226
idx = 227
temp1=  301
idx= 227
idx = 228
temp1=  302
idx= 228
idx = 229
temp1=  303
idx= 229
idx = 230
temp1=  304
idx= 230
idx = 231
temp1=  305
idx= 231
idx = 232
temp1=  306
idx= 232
idx = 233
temp1=  307
idx= 233
idx = 234
temp1=  308
idx= 234
idx = 235
temp1=  309
idx= 235
idx = 236
temp1=  311
idx= 236
idx = 237
temp1=  312
idx= 237
idx = 238
temp1=  313
idx= 238
idx = 239
temp1=  314
idx= 239
idx = 240
idx = 240
temp1=  315
idx= 240
idx = 241
temp1=  316
idx= 241
idx = 242
temp1=  317
idx= 242
idx = 243
temp1=  318
idx= 243
idx = 244
temp1=  319
idx= 244
idx = 245
temp1=  320
idx= 245
idx = 246
temp1=  321
idx= 246
idx = 247
temp1=  322
idx= 247
idx = 248
temp1=  323
idx= 248
idx = 249
temp1=  324
idx= 249
idx = 250
temp1=  325
idx= 250
idx = 251
temp1=  326
idx= 251
idx = 252
temp1=  327
idx= 252
idx = 253
temp1=  330
idx= 253
idx = 254
temp1=  331
idx= 254
idx = 255
```

```
IOPub message rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_msg_rate_limit`.

Current values:
NotebookApp.iopub_msg_rate_limit=1000.0 (msgs/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

In [99]:

```python
c_0=[]
c_1=[]
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])


print(len(c_0))
print(len(c_1))
print(len(label))
```

```
366
366
366
```

```
366
```

In [101]:

```
a=X_train['clean_subcategories'].unique()
a=a[0:366]
len(a)
```

Out[101]:

```
366
```

In [102]:

```
#Creating A Response Table
res_table_subcat=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table_subcat['prob_0']=c_0
res_table_subcat['prob_1']=c_1
#res_table['label']=label
res_table_subcat['categories']=a
```

In [103]:

```
res_table_subcat.head()
```

Out[103]:

	prob_0	prob_1	categories
appliedsciences mathematics	0.187160	0.812840	appliedsciences mathematics
civicsgovernment healthwellness	0.157895	0.842105	civicsgovernment healthwellness
literaturewriting specialneeds	0.125000	0.875000	literaturewriting specialneeds
literaturewriting	0.167785	0.832215	literaturewriting
environmentalscience mathematics	0.112903	0.887097	environmentalscience mathematics

Training based on response_table

Train Data

In [105]:

```
train_coded_subcat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [106]:

```
temp_0=[]
temp_1=[]
for cat in X_train["clean_subcategories"].values:
    if cat in res_table_subcat["categories"].values:
        temp_0.append(res_table_subcat.loc[cat,"prob_0"])
        temp_1.append(res_table_subcat.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [107]:

```
train_coded_subcat["prob_0"]=temp_0
train_coded_subcat["prob_1"]=temp_1
```

In [108]:

```
train_coded_subcat.shape
```

Out[108]:

```
(40401, 2)
```

CV data

In [109]:

```
cv_coded_subcat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [110]:

```
temp_0=[]
temp_1=[]
for cat in X_cv["clean_subcategories"].values:
    if cat in res_table_subcat["categories"].values:
        temp_0.append(res_table_subcat.loc[cat,"prob_0"])
        temp_1.append(res_table_subcat.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [111]:

```
cv_coded_subcat["prob_0"]=temp_0
cv_coded_subcat["prob_1"]=temp_1
```

In [112]:

```
cv_coded_subcat.shape
```

Out[112]:

```
(19899, 2)
```

Test data

In [113]:

```
test_coded_subcat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [114]:

```
temp_0=[]
temp_1=[]
for cat in X_test["clean_subcategories"].values:
    if cat in res_table_subcat["categories"].values:
        temp_0.append(res_table_subcat.loc[cat,"prob_0"])
        temp_1.append(res_table_subcat.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [115]:

```
test_coded_subcat["prob_0"]=temp_0
test_coded_subcat["prob_1"]=temp_1
```

In [116]:

```
test_coded_subcat.shape
```

Out[116]:

```
(29700, 2)
```

Teacher_Prefix

In [117]:

```
init_data=pd.DataFrame(columns=['categories','label'])
init_data['categories']=X_train['preprocessed_prefix']
init_data['label']=y_train
```

In [118]:

```
print(init_data.head())
print(init_data.shape)
```

```
      categories  label
48062        mrs      1
81103        mrs      1
19797         ms      1
86367        mrs      1
64405        mrs      1
(40401, 2)
```

In [119]:

```
#how to calculate conditional probability python pandas -
>https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
n-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
```

In [120]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div
(cond_prob , axis=0, level='categories'),columns=['probability'])
```

In [121]:

```
print(encoded_cat.head())
print(encoded_cat.tail())
```

```
                  probability
label categories             
0     mr             0.154425
      mrs            0.144766
      ms             0.160987
      teacher        0.181818
1     dr             1.000000
                  probability
label categories             
1     mr             0.845575
      mrs            0.855234
      ms             0.839013
      nan            1.000000
      teacher        0.818182
```

In [122]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[122]:

```
(10, 3)
```

In [123]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
print(cat_0.head())
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label categories  probability
0      0         mr     0.154425
1      0        mrs     0.144766
2      0         ms     0.160987
3      0    teacher     0.181818
(4, 3)
   label categories  probability
4      1         dr     1.000000
5      1         mr     0.845575
6      1        mrs     0.855234
7      1         ms     0.839013
8      1        nan     1.000000
(6, 3)
```

In [124]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [125]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  1
idx= 0
idx = 1
temp1=  2
idx= 1
idx = 2
temp1=  3
idx= 2
idx = 3
temp1=  5
idx= 3
idx = 4
idx = 5
```

In [126]:

```
c_0=[]
c_1=[]
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])
```

```
print(len(c_0))
print(len(c_1))
print(len(label))
```

```
6
6
6
```

In [127]:

```
a=X_train['preprocessed_prefix'].unique()
len(a)
```

Out[127]:

```
6
```

In [128]:

```
#Creating A Response Table
res_table_prefix=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table_prefix['prob_0']=c_0
res_table_prefix['prob_1']=c_1
#res_table['label']=label
res_table_prefix['categories']=a
```

In [129]:

```
res_table_prefix.head()
```

Out[129]:

	prob_0	prob_1	categories
mrs	0.154425	0.845575	mrs
ms	0.144766	0.855234	ms
mr	0.160987	0.839013	mr
teacher	0.181818	0.818182	teacher
nan	0.000000	1.000000	nan

Training Based on Response Table

Train Data

In [130]:

```
train_coded_prefix=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [131]:

```
temp_0=[]
temp_1=[]
for cat in X_train["preprocessed_prefix"].values:
    if cat in res_table_prefix["categories"].values:
        temp_0.append(res_table_prefix.loc[cat,"prob_0"])
        temp_1.append(res_table_prefix.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [132]:

```
train_coded_prefix["prob_0"]=temp_0
```

```
train_coded_prefix["prob_0"]=temp_0
train_coded_prefix["prob_1"]=temp_1
```

In [133]:

```
train_coded_prefix.shape
```

Out[133]:

```
(40401, 2)
```

CV data

In [134]:

```
cv_coded_prefix=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [135]:

```
temp_0=[]
temp_1=[]
for cat in X_cv["preprocessed_prefix"].values:
    if cat in res_table_prefix["categories"].values:
        temp_0.append(res_table_prefix.loc[cat,"prob_0"])
        temp_1.append(res_table_prefix.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [136]:

```
cv_coded_prefix["prob_0"]=temp_0
cv_coded_prefix["prob_1"]=temp_1
```

In [137]:

```
cv_coded_prefix.shape
```

Out[137]:

```
(19899, 2)
```

Test Data

In [138]:

```
test_coded_prefix=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [139]:

```
temp_0=[]
temp_1=[]
for cat in X_test["preprocessed_prefix"].values:
    if cat in res_table_prefix["categories"].values:
        temp_0.append(res_table_prefix.loc[cat,"prob_0"])
        temp_1.append(res_table_prefix.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [140]:

```
test_coded_prefix["prob_0"]=temp_0
test_coded_prefix["prob_1"]=temp_1
```

In [141]:

In [141]:

```
test_coded_prefix.shape
```

Out[141]:

```
(29700, 2)
```

Grade Category

In [142]:

```
init_data=pd.DataFrame(columns=['categories','label'])


init_data['categories']=X_train['preprocessed_grade']
init_data['label']=y_train
```

In [143]:

```
print(init_data.head())
print(init_data.shape)
```

```
          categories  label
48062  Grades_PreK_2      1
81103    Grades_9_12      1
19797  Grades_PreK_2      1
86367     Grades_6_8      1
64405     Grades_6_8      1
(40401, 2)
```

In [144]:

```
#how to calculate conditional probability python pandas -
>https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
n-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
```

In [145]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div
(cond_prob , axis=0, level='categories'),columns=['probability'])
```

In [146]:

```
print(encoded_cat.head())
init_data['categories']=X_train['preprocessed_grade']
init_data['label']=y_train
print(encoded_cat.tail())
```

```
                     probability
label categories
0     Grades_3_5        0.144337
      Grades_6_8        0.160816
      Grades_9_12       0.162143
      Grades_PreK_2     0.153136
1     Grades_3_5        0.855663
                     probability
label categories
0     Grades_PreK_2     0.153136
1     Grades_3_5        0.855663
      Grades_6_8        0.839184
      Grades_9_12       0.837857
      Grades_PreK_2     0.846864
```

In [147]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[147]:

```
(8, 3)
```

In [148]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label     categories  probability
0      0     Grades_3_5     0.144337
1      0     Grades_6_8     0.160816
2      0    Grades_9_12     0.162143
3      0  Grades_PreK_2     0.153136
(4, 3)
   label     categories  probability
4      1     Grades_3_5     0.855663
5      1     Grades_6_8     0.839184
6      1    Grades_9_12     0.837857
7      1  Grades_PreK_2     0.846864
(4, 3)
```

In [149]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [150]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  0
idx= 0
idx = 1
temp1=  1
idx= 1
idx = 2
temp1=  2
idx= 2
idx = 3
temp1=  3
idx= 3
```

In [151]:

```
c_0=[]
c_1=[]
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])


print(len(c_0))
print(len(c_1))
print(len(label))
```

```
4
4
4
```

In [152]:

```
a=X_train['preprocessed_grade'].unique()
len(a)
```

Out[152]:

```
4
```

In [153]:

```
#Creating A Response Table
res_table_grade=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table_grade['prob_0']=c_0
res_table_grade['prob_1']=c_1
#res_table['label']=label
res_table_grade['categories']=a
```

In [154]:

```
res_table_grade.head()
```

Out[154]:

	prob_0	prob_1	categories
Grades_PreK_2	0.144337	0.855663	Grades_PreK_2
Grades_9_12	0.160816	0.839184	Grades_9_12
Grades_6_8	0.162143	0.837857	Grades_6_8
Grades_3_5	0.153136	0.846864	Grades_3_5

Training Based on Response Table

Train Data

In [155]:

```
train_coded_grade=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [156]:

```
temp_0=[]
temp_1=[]
for cat in X_train['preprocessed_grade'].values:
    if cat in res_table_grade["categories"].values:
        temp_0.append(res_table_grade.loc[cat,"prob_0"])
        temp_1.append(res_table_grade.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [157]:

```
train_coded_grade["prob_0"]=temp_0
train_coded_grade["prob_1"]=temp_1
```

In [158]:

```
train_coded_grade.shape
```

Out[158]:

```
(40401, 2)
```

CV data

In [159]:

```
cv_coded_grade=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [160]:

```
temp_0=[]
temp_1=[]
for cat in X_cv['preprocessed_grade'].values:
    if cat in res_table_grade["categories"].values:
        temp_0.append(res_table_grade.loc[cat,"prob_0"])
        temp_1.append(res_table_grade.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [161]:

```
cv_coded_grade["prob_0"]=temp_0
cv_coded_grade["prob_1"]=temp_1
```

In [162]:

```
cv_coded_grade.shape
```

Out[162]:

```
(19899, 2)
```

Test Data

In [163]:

```
test_coded_grade=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [164]:

```
temp_0=[]
temp_1=[]
for cat in X_test['preprocessed_grade'].values:
    if cat in res_table_grade["categories"].values:
        temp_0.append(res_table_grade.loc[cat,"prob_0"])
        temp_1.append(res_table_grade.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [165]:

```
test_coded_grade["prob_0"]=temp_0
test_coded_grade["prob_1"]=temp_1
```

In [166]:

```
test_coded_grade.shape
```

Out[166]:

```
(29700, 2)
```

School_State Feature

In [167]:

```
init_data=pd.DataFrame(columns=['categories','label'])


init_data['categories']=X_train['school_state']
init_data['label']=y_train



print(init_data.head())
print(init_data.shape)
```

```
      categories  label
48062         UT      1
81103         TN      1
19797         NY      1
86367         OH      1
64405         CA      1
(40401, 2)
```

In [168]:

```
#how to calculate conditional probability python pandas -
>https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
n-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
```

In [169]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div
(cond_prob , axis=0, level='categories'),columns=['probability'])



print(encoded_cat.head())
print(encoded_cat.tail())
```

```
                  probability
label categories             
0     AK             0.142857
      AL             0.137821
      AR             0.170213
      AZ             0.156373
      CA             0.145844
                  probability
label categories             
1     VT             0.794872
      WA             0.872642
      WI             0.838663
      WV             0.843243
      WY             0.823529
```

In [171]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[171]:

```
(102, 3)
```

In [172]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label categories  probability
0      0         AK     0.142857
1      0         AL     0.137821
2      0         AR     0.170213
3      0         AZ     0.156373
4      0         CA     0.145844
(51, 3)
    label categories  probability
51      1         AK     0.857143
52      1         AL     0.862179
53      1         AR     0.829787
54      1         AZ     0.843627
55      1         CA     0.854156
(51, 3)
```

In [173]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [174]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  0
idx= 0
idx = 1
temp1=  1
idx= 1
idx = 2
temp1=  2
idx= 2
idx = 3
temp1=  3
```

```
idx= 3
idx = 4
temp1=  4
idx= 4
idx = 5
temp1=  5
idx= 5
idx = 6
temp1=  6
idx= 6
idx = 7
temp1=  7
idx= 7
idx = 8
temp1=  8
idx= 8
idx = 9
temp1=  9
idx= 9
idx = 10
temp1=  10
idx= 10
idx = 11
temp1=  11
idx= 11
idx = 12
temp1=  12
idx= 12
idx = 13
temp1=  13
idx= 13
idx = 14
temp1=  14
idx= 14
idx = 15
temp1=  15
idx= 15
idx = 16
temp1=  16
idx= 16
idx = 17
temp1=  17
idx= 17
idx = 18
temp1=  18
idx= 18
idx = 19
temp1=  19
idx= 19
idx = 20
temp1=  20
idx= 20
idx = 21
temp1=  21
idx= 21
idx = 22
temp1=  22
idx= 22
idx = 23
temp1=  23
idx= 23
idx = 24
temp1=  24
idx= 24
idx = 25
temp1=  25
idx= 25
idx = 26
temp1=  26
idx= 26
idx = 27
temp1=  27
idx= 27
idx = 28
temp1=  28
idx= 28
idx = 29
```

```
temp1=  29
idx= 29
idx = 30
temp1=  30
idx= 30
idx = 31
temp1=  31
idx= 31
idx = 32
temp1=  32
idx= 32
idx = 33
temp1=  33
idx= 33
idx = 34
temp1=  34
idx= 34
idx = 35
temp1=  35
idx= 35
idx = 36
temp1=  36
idx= 36
idx = 37
temp1=  37
idx= 37
idx = 38
temp1=  38
idx= 38
idx = 39
temp1=  39
idx= 39
idx = 40
temp1=  40
idx= 40
idx = 41
temp1=  41
idx= 41
idx = 42
temp1=  42
idx= 42
idx = 43
temp1=  43
idx= 43
idx = 44
temp1=  44
idx= 44
idx = 45
temp1=  45
idx= 45
idx = 46
temp1=  46
idx= 46
idx = 47
temp1=  47
idx= 47
idx = 48
temp1=  48
idx= 48
idx = 49
temp1=  49
idx= 49
idx = 50
temp1=  50
idx= 50
```

In [175]:

```
c_0=[]
c_1=[]
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])
```

```
print(len(c_0))
print(len(c_1))
print(len(label))
```

```
51
51
51
```

In [176]:

```
a=X_train['school_state'].unique()
len(a)
```

Out[176]:

```
51
```

In [177]:

```
#Creating A Response Table
res_table_state=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table_state['prob_0']=c_0
res_table_state['prob_1']=c_1
#res_table['label']=label
res_table_state['categories']=a
```

In [178]:

```
res_table_state.head()
```

Out[178]:

	prob_0	prob_1	categories
UT	0.142857	0.857143	UT
TN	0.137821	0.862179	TN
NY	0.170213	0.829787	NY
OH	0.156373	0.843627	OH
CA	0.145844	0.854156	CA

Training Based on Response Table

Train Data

In [179]:

```
train_coded_state=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [180]:

```
temp_0=[]
temp_1=[]
for cat in X_train["school_state"].values:
    if cat in res_table_state["categories"].values:
        temp_0.append(res_table_state.loc[cat,"prob_0"])
        temp_1.append(res_table_state.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [181]:

```
train_coded_state["prob_0"]=temp_0
```

```
train_coded_state["prob_1"]=temp_1
```

In [182]:

```
train_coded_state.shape
```

Out[182]:

```
(40401, 2)
```

CV Data

In [183]:

```
cv_coded_state=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [184]:

```
temp_0=[]
temp_1=[]
for cat in X_cv["school_state"].values:
    if cat in res_table_state["categories"].values:
        temp_0.append(res_table_state.loc[cat,"prob_0"])
        temp_1.append(res_table_state.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [185]:

```
cv_coded_state["prob_0"]=temp_0
cv_coded_state["prob_1"]=temp_1
```

In [186]:

```
cv_coded_state.shape
```

Out[186]:

```
(19899, 2)
```

Test Data

In [187]:

```
test_coded_state=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [188]:

```
temp_0=[]
temp_1=[]
for cat in X_test["school_state"].values:
    if cat in res_table_state["categories"].values:
        temp_0.append(res_table_state.loc[cat,"prob_0"])
        temp_1.append(res_table_state.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [189]:

```
test_coded_state["prob_0"]=temp_0
test_coded_state["prob_1"]=temp_1
```

In [190]:

```
test_coded_state.shape
```

Out[190]:

```
(29700, 2)
```

Vectorizing Text Data

Bag of words(BoW)

Preprocessed Essay

In [191]:

```
model_essay_bow =  CountVectorizer(min_df=10)
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  (40401, 11081)
==================================================
Shape of matrix  (19899, 11081)
==================================================
Shape of matrix  (29700, 11081)
```

Preprocessed Title

In [192]:

```
model_title_bow =  CountVectorizer(min_df=10)
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  (40401, 1750)
==================================================
Shape of matrix  (19899, 1750)
==================================================
Shape of matrix  (29700, 1750)
```

Tf-idf vectorizer

Tf-idf of Project_Essays

In [193]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
model_essay_tfidf = TfidfVectorizer(min_df=10)
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("Shape of matrix ",train_tfidf_essay.shape)
print("="*50)
cv_tfidf_essay=model_essay_tfidf.transform(X_cv["preprocessed_essays"]) #tfidf of CV
print("Shape of matrix ",cv_tfidf_essay.shape)
print("="*50)
test_tfidf_essay = model_essay_tfidf.transform(X_test["preprocessed_essays"]) #tfidf of Test
print("Shape of matrix ",test_tfidf_essay.shape)
```

```
Shape of matrix  (40401, 11081)
==================================================
Shape of matrix  (19899, 11081)
==================================================
Shape of matrix  (29700, 11081)
```

Tf-idf of Project_Title

In [194]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
model_title_tfidf = TfidfVectorizer(min_df=10)
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"]) #tfidf of CV
print("Shape of matrix ",cv_tfidf_title.shape)
print("="*50)
test_tfidf_title = model_title_tfidf.transform(X_test["preprocessed_title"]) #tfidf of Test
print("Shape of matrix ",test_tfidf_title.shape)
```

```
Shape of matrix  (40401, 1750)
==================================================
Shape of matrix  (19899, 1750)
==================================================
Shape of matrix  (29700, 1750)
```

Making numerical features hstack compatible

Train

In [195]:

```
price_train=X_train['price'].values.reshape(1,-1)
print(price_train.shape)
```

```
(1, 40401)
```

In [196]:

```
price_train=price_train.reshape(-1,1)
print(price_train.shape)
```

```
(40401, 1)
```

In [197]:

```
quantity_train=X_train['quantity'].values.reshape(1,-1)
print(quantity_train.shape)
```

```
(1, 40401)
```

In [198]:

```
quantity_train=quantity_train.reshape(-1,1)
```

```
print(quantity_train.shape)
```

```
(40401, 1)
```

In [199]:

```
tnp_train=X_train["teacher_number_of_previously_posted_projects"].values.reshape(1,-1)
print(tnp_train.shape)
```

```
(1, 40401)
```

In [200]:

```
tnp_train=tnp_train.reshape(-1,1)
print(tnp_train.shape)
```

```
(40401, 1)
```

Cross-Validation

In [201]:

```
price_cv=X_cv['price'].values.reshape(1,-1)
print(price_cv.shape)
```

```
(1, 19899)
```

In [202]:

```
price_cv=price_cv.reshape(-1,1)
print(price_cv.shape)
```

```
(19899, 1)
```

In [203]:

```
quantity_cv=X_cv['quantity'].values.reshape(1,-1)
print(quantity_cv.shape)
```

```
(1, 19899)
```

In [204]:

```
quantity_cv=quantity_cv.reshape(-1,1)
print(quantity_cv.shape)
```

```
(19899, 1)
```

In [205]:

```
tnp_cv=X_cv["teacher_number_of_previously_posted_projects"].values.reshape(1,-1)
print(tnp_cv.shape)
```

```
(1, 19899)
```

In [206]:

```
tnp_cv=tnp_cv.reshape(-1,1)
print(tnp_cv.shape)
```

```
(19899, 1)
```

Test

In [207]:

```
price_test=X_test['price'].values.reshape(1,-1)
print(price_test.shape)
```

```
(1, 29700)
```

In [208]:

```
price_test=price_test.reshape(-1,1)
print(price_test.shape)
```

```
(29700, 1)
```

In [209]:

```
quantity_test=X_test['quantity'].values.reshape(1,-1)
print(quantity_test.shape)
```

```
(1, 29700)
```

In [210]:

```
quantity_test=quantity_test.reshape(-1,1)
print(quantity_test.shape)
```

```
(29700, 1)
```

In [211]:

```
tnp_test=X_test["teacher_number_of_previously_posted_projects"].values.reshape(1,-1)
print(tnp_test.shape)
```

```
(1, 29700)
```

In [212]:

```
tnp_test=tnp_test.reshape(-1,1)
print(tnp_test.shape)
```

```
(29700, 1)
```

Applying Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees

Set 1: Categorical Features,Numerical Features+Preprocessed Essay(BOW)+Preprocessed Title(BOW)

In [213]:

```
from scipy.sparse import hstack
X_tr_1=hstack((train_coded_cat,train_coded_subcat,train_coded_prefix,train_coded_grade,train_coded_
state,price_train,quantity_train,tnp_train,train_bow_essay,train_bow_title)).tocsr()

X_cv_1=hstack((cv_coded_cat,cv_coded_subcat,cv_coded_prefix,cv_coded_grade,cv_coded_state,price_cv
,quantity_cv,tnp_cv,cv_bow_essay,cv_bow_title)).tocsr()
```

```
,quantity_cv,tnp_cv,cv_bow_essay,cv_bow_title)).tocsr()

X_te_1=hstack((test_coded_cat,test_coded_subcat,test_coded_prefix,test_coded_grade
,test_coded_state,price_test,quantity_test,tnp_test,test_bow_essay,test_bow_title)).tocsr()
```

In [214]:

```
#checking the final matrix are of same dimension or not
print(X_tr_1.shape,y_train.shape)
print("="*50)
print(X_cv_1.shape,y_cv.shape)
print("="*50)
print(X_te_1.shape,y_test.shape)
```

```
(40401, 12844) (40401,)
==================================================
(19899, 12844) (19899,)
==================================================
(29700, 12844) (29700,)
```

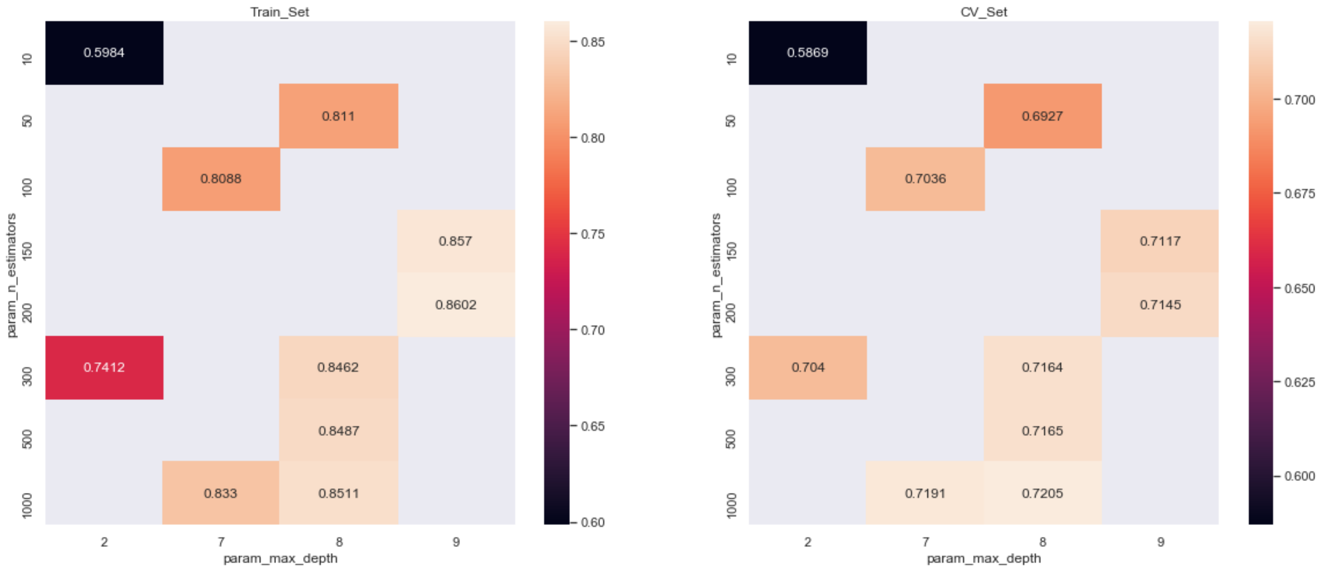
finding best Hyperparameters Using RandomizedSearchCV

In [215]:

```
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(class_weight='balanced')
parameters={'n_estimators':[10, 50, 100, 150, 200, 300, 500, 1000], 'max_depth':[2, 3, 4, 5, 6, 7, 8
, 9, 10]}

clf=RandomizedSearchCV(rf,parameters, cv=3, scoring='roc_auc', return_train_score=True)

set1=clf.fit(X_tr_1,y_train)
```

In [216]:

```
import seaborn as sns
sns.set()
df1=pd.DataFrame(clf.cv_results_).groupby(['param_n_estimators', 'param_max_depth']).max().unstack
()[['mean_test_score', 'mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,8))
sns.heatmap(df1.mean_train_score,annot=True, fmt='.4g', ax=ax[0])
sns.heatmap(df1.mean_test_score,annot=True, fmt='.4g', ax=ax[1])
ax[0].set_title("Train_Set")
ax[1].set_title("CV_Set")
plt.show()
```



```
In [217]:
```

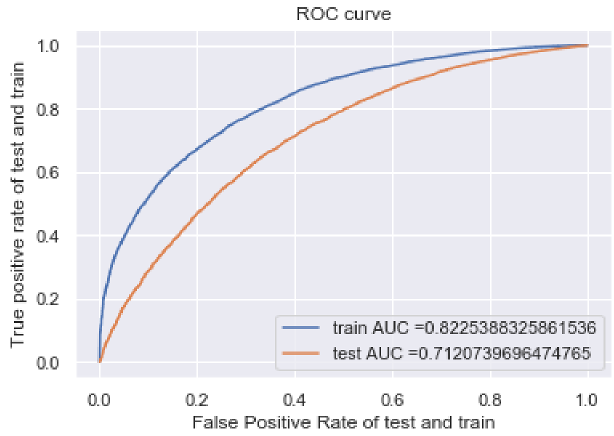
```
print(clf.best_estimator_)

print(clf.score(X_tr_1,y_train))
print(clf.score(X_cv_1,y_cv))
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=8, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=1000, n_jobs=None, oob_score=False,
            random_state=None, verbose=0, warm_start=False)
0.8239454781191868
0.7221090417773771
```

Testing on Test Data(using our max_depth=8 and n_estimators=1000)

```
In [218]:
```

```
rf = RandomForestClassifier(n_estimators=1000, max_depth=8,class_weight='balanced')

rf.fit(X_tr_1, y_train)
train_predict=rf.predict_proba(X_tr_1)[:,1]
test_predict= rf.predict_proba(X_te_1)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,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(True)
plt.show()
```



Confusion Matrix

```
In [219]:
```

```
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   1.642718
1  0.0  0.000029   0.642718
2  0.0  0.037023   0.568427
(8123, 3)
```

In [220]:

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

In [221]:

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

In [222]:

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

df.head(3)
```

Out[222]:

	fpr	tpr	threshold	Specificty	Value
2937	0.261541	0.742095	0.499872	0.738459	0.548007
2967	0.263979	0.744547	0.499703	0.736021	0.548002
2981	0.265767	0.746358	0.499551	0.734233	0.548000

In [223]:

```
index = df.Value.argmax()
```

In [224]:

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

```
0.49987164550772856
```

In [225]:

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

```
[0.]
```

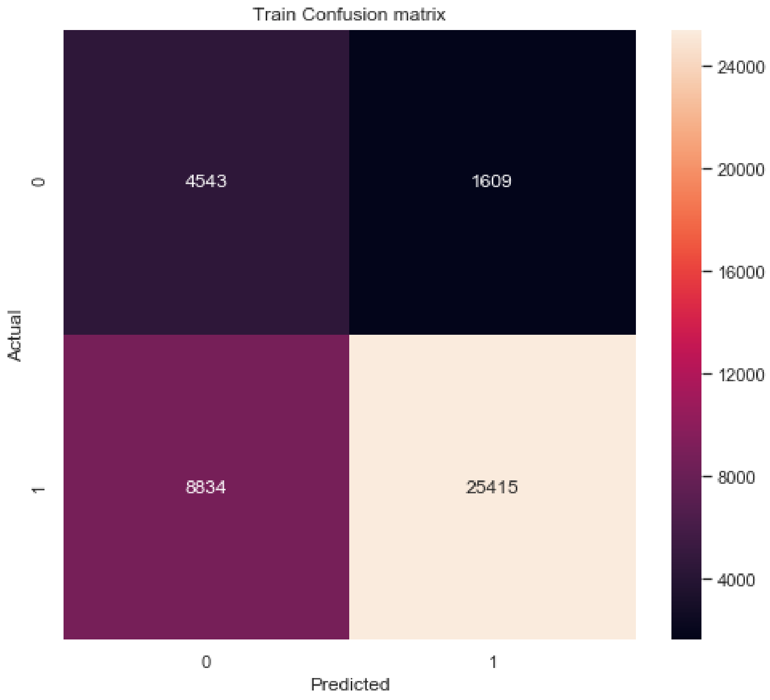
In [226]:

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

#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")
```

```
Threshold 0.49987164550772856
confusion matrix
[[ 4543  1609]
 [ 8834 25415]]
```

Out[226]:

```
Text(0.5, 39.5, 'Predicted')
```



Test Data

In [227]:

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

```
[0.]
```

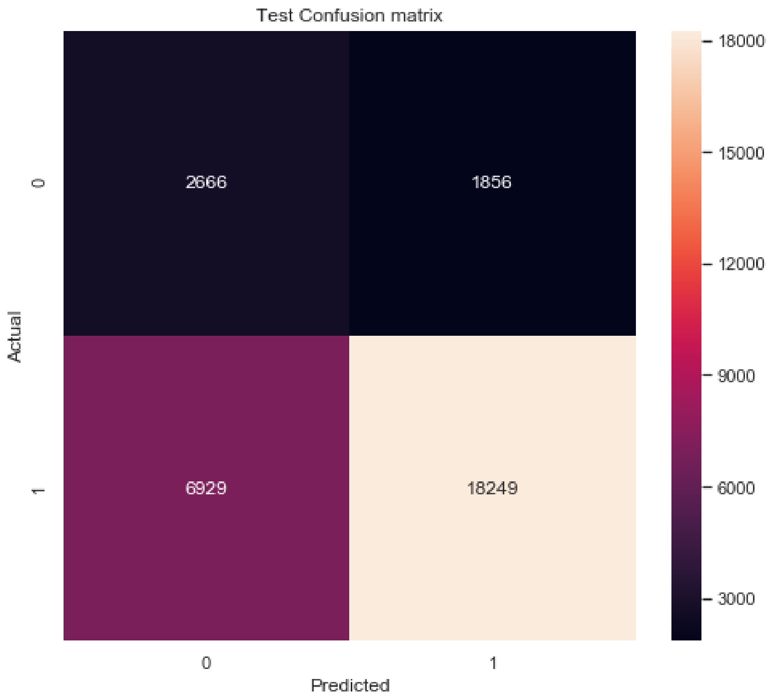
In [228]:

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

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

#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")
```

```
Threshold 0.49987164550772856
Test confusion matrix
[[ 2666  1856]
 [ 6929 18249]]
```

Out[228]:

```
Text(0.5, 39.5, 'Predicted')
```



Set 2: Categorical Features,Numerical Features+Preprocessed Essay(tf-idf)+Preprocessed Title(tf-idf)

In [229]:

```
from scipy.sparse import hstack
X_tr_2=hstack((train_coded_cat,train_coded_subcat,train_coded_prefix,train_coded_grade,train_coded_
state,price_train,quantity_train,tnp_train,train_tfidf_title,train_tfidf_essay)).tocsr()

X_cv_2=hstack((cv_coded_cat,cv_coded_subcat,cv_coded_prefix,cv_coded_grade,cv_coded_state,price_cv
,quantity_cv,tnp_cv,cv_tfidf_essay,cv_tfidf_title)).tocsr()

X_te_2=hstack((test_coded_cat,test_coded_subcat,test_coded_prefix,test_coded_grade
,test_coded_state,price_test,quantity_test,tnp_test,test_tfidf_essay,test_tfidf_title)).tocsr()
```

In [230]:

```
#checking the final matrix are of same dimension or not
print(X_tr_2.shape,y_train.shape)
print("="*50)
print(X_cv_2.shape,y_cv.shape)
print("="*50)
print(X_te_2.shape,y_test.shape)
```

```
(40401, 12844) (40401,)
==================================================
(19899, 12844) (19899,)
==================================================
(29700, 12844) (29700,)
```

finding best Hyperparameters using RandomizedSearchCV

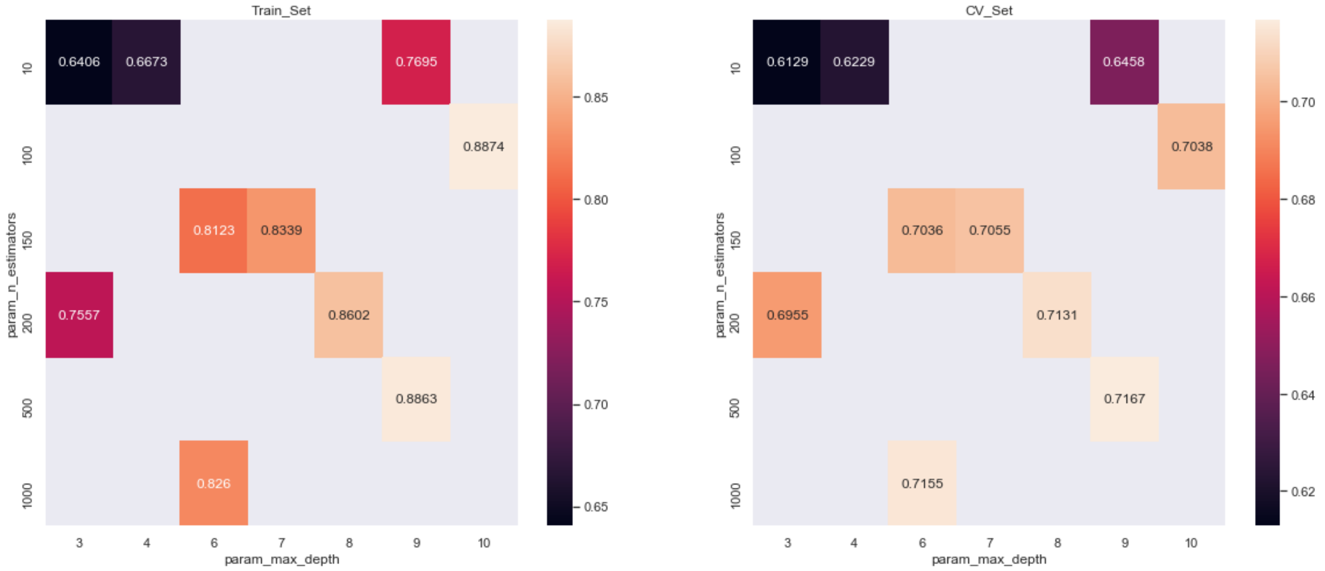
In [231]:

```
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(class_weight='balanced')
parameters={'n_estimators':[10, 50, 100, 150, 200, 300, 500, 1000], 'max_depth':[2, 3, 4, 5, 6, 7, 8
, 9, 10]}

clf=RandomizedSearchCV(rf,parameters, cv=3, scoring='roc_auc', return_train_score=True)
```

```
set2=clf.fit(X_tr_2,y_train)
```

In [232]:

```
import seaborn as sns
sns.set()
df2=pd.DataFrame(clf.cv_results_).groupby(['param_n_estimators', 'param_max_depth']).max().unstack
()[['mean_test_score', 'mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,8))
sns.heatmap(df2.mean_train_score,annot=True, fmt='.4g', ax=ax[0])
sns.heatmap(df2.mean_test_score,annot=True, fmt='.4g', ax=ax[1])
ax[0].set_title("Train_Set")
ax[1].set_title("CV_Set")
plt.show()
```



In [233]:

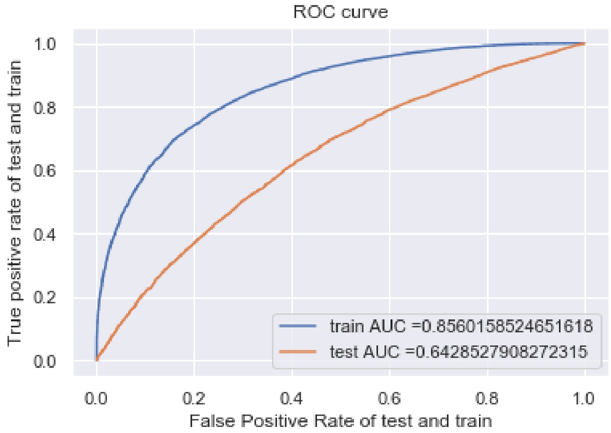
```
print(clf.best_estimator_)

print(clf.score(X_tr_2,y_train))
print(clf.score(X_cv_2,y_cv))
```

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=9, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=500, n_jobs=None, oob_score=False,
            random_state=None, verbose=0, warm_start=False)
0.8548887443905513
0.6450535450130466
```

Testing on Test Data(using our max_depth=9 and n_estimators=500)

In [234]:

```
rf = RandomForestClassifier(max_depth=9, n_estimators=500 ,class_weight='balanced')

rf.fit(X_tr_2, y_train)
train_predict=rf.predict_proba(X_tr_2)[:,1]
test_predict= rf.predict_proba(X_te_2)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train,train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test,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(True)
plt.show()
```



Confusion Matrix

In [235]:

```
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   1.647235
1  0.0  0.000029   0.647235
2  0.0  0.019125   0.586954
(7401, 3)
```

In [236]:

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

In [237]:

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

In [238]:

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

df.head(3)
```

Out[238]:

	fpr	tpr	threshold	Specificty	Value
2636	0.231632	0.776052	0.505312	0.768368	0.596293
2672	0.235208	0.779672	0.505030	0.764792	0.596287
2638	0.231795	0.776081	0.505310	0.768205	0.596190

In [239]:

```
index = df.Value.argmax()
```

In [240]:

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

```
0.5053116280560193
```

In [241]:

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

```
[0.]
```

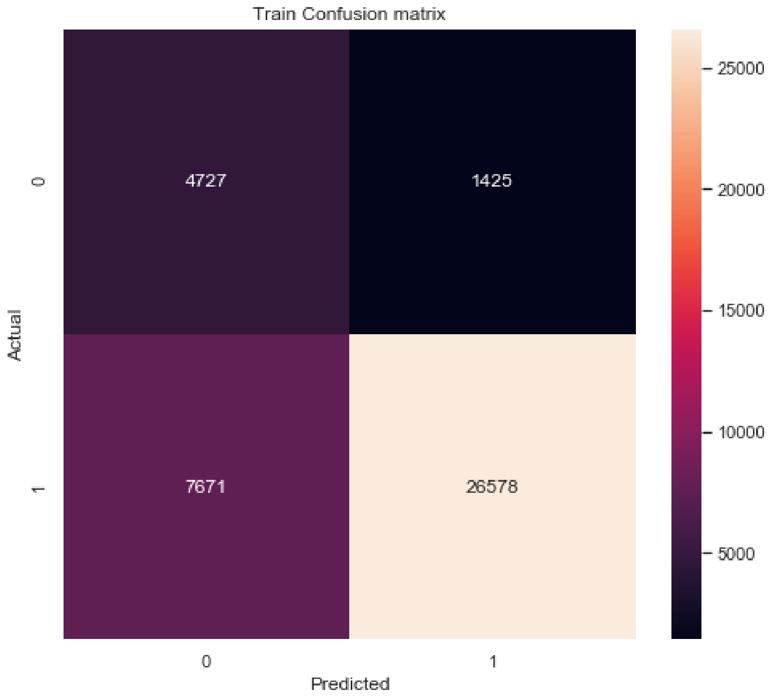
In [242]:

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

#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")
```

```
Threshold 0.5053116280560193
confusion matrix
[[ 4727  1425]
 [ 7671 26578]]
```

Out[242]:

```
Text(0.5, 39.5, 'Predicted')
```



Test Data

In [243]:

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

```
[0.]
```

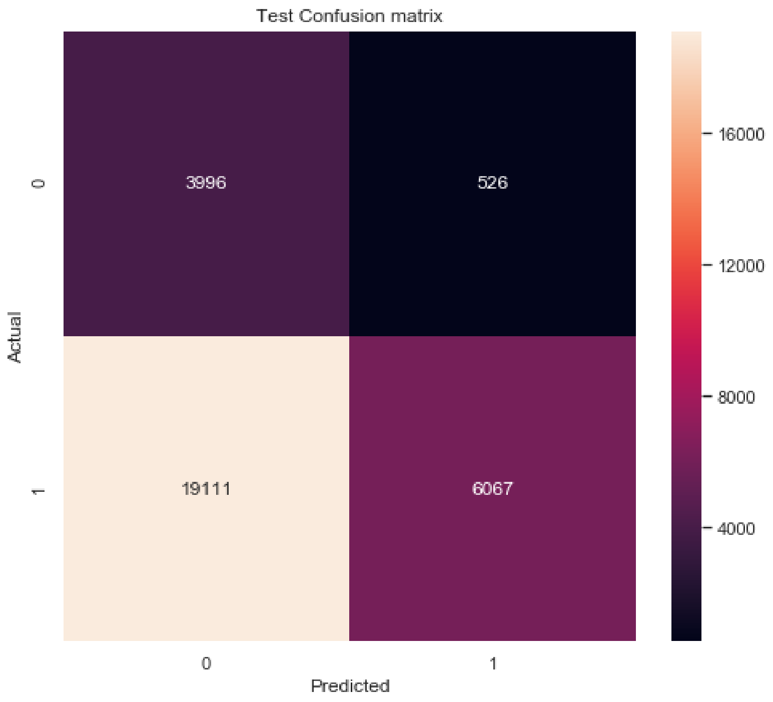
In [244]:

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

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


#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")
```

```
Threshold 0.5053116280560193
Test confusion matrix
[[ 3996   526]
 [19111  6067]]
```

Out[244]:

```
Text(0.5, 39.5, 'Predicted')
```



Now for set3 and set4 and XGboost we will be taking less data to reduce training time

Feature Preprocessing

Preprocessing of project_subject_categories

Train_Data1

In [246]:

```
print(train_data1.project_subject_categories[0:5])
print("*"*50)
categories1=list(train_data1["project_subject_categories"].values)#created a list of the values in
the project_subject_categories
print(categories1[0:5])
```

```
0         Math & Science
1          Special Needs
2    Literacy & Language
3       Applied Learning
4    Literacy & Language
Name: project_subject_categories, dtype: object
**************************************************
['Math & Science', 'Special Needs', 'Literacy & Language', 'Applied Learning', 'Literacy &
Language']
```

In [247]:

```
clean_cat1=[]
for i in categories1: #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_cat1.append(temp.strip())
    #showing the result
print(clean_cat1[0:5])
```

```
['mathscience', 'specialneeds', 'literacylanguage', 'appliedlearning', 'literacylanguage']
```

In [248]:

```
train_data1['clean_categories']=clean_cat1 #creating a new column as clean_categories
train_data1.drop(['project_subject_categories'], axis=1,inplace=True) #dropping the subject catego
ry
```

In [249]:

```
# 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_data1['clean_categories'].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/
cat_dict1 = dict(my_counter1)
sorted_cat_dict1 = dict(sorted(cat_dict1.items(), key=lambda kv:(kv[1] ,kv[0])))
```

```
{'mathscience': 10767, 'specialneeds': 3324, 'literacylanguage': 14494, 'appliedlearning': 3293, '
historycivics': 1623, 'musicarts': 2416, 'healthsports': 5693}
```

```
historycivics': 1623, 'musicarts': 2416, 'healthsports': 5693,
==================================================
[('appliedlearning', 3293), ('healthsports', 5693), ('historycivics', 1623), ('literacylanguage',
14494), ('mathscience', 10767), ('musicarts', 2416), ('specialneeds', 3324)]
```

Preprocessing of project_subject_subcategories

In [251]:

```
print(train_data1.project_subject_subcategories[0:5])
print("*"*50)
categories1=list(train_data1["project_subject_subcategories"].values)#created a list of the values
in the project_subject_categories
print(categories1[0:5])
```

```
0    Applied Sciences, Health & Life Science
1                              Special Needs
2                                   Literacy
3                          Early Development
4                                   Literacy
Name: project_subject_subcategories, dtype: object
**************************************************
['Applied Sciences, Health & Life Science', 'Special Needs', 'Literacy', 'Early Development', 'Lit
eracy']
```

In [252]:

```
#Refer ->https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
#Refer for documentation ->https://www.programiz.com/python-programming/methods/string/strip
subcategories1 = list(train_data1['project_subject_subcategories'].values) #creating a list of
all the values in project subject categories
clean_subcat1=[]
for i in subcategories1: #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_subcat1.append(temp.strip())
    #showing the result
print(clean_subcat1[0:5])
```

```
['appliedsciences healthlifescience', 'specialneeds', 'literacy', 'earlydevelopment', 'literacy']
```

In [253]:

```
train_data1['clean_subcategories']=clean_subcat1 #creating a new column as clean_categories
train_data1.drop(['project_subject_subcategories'], axis=1,inplace=True) #dropping the subject cat
egory
```

In [254]:

```
# 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_1 = Counter()
for word in train_data1['clean_subcategories'].values:
    my_counter_1.update(word.split())

print(dict(my_counter_1)) #printing the dictionary
sortd_1=sorted(my_counter_1.items()) #with sorted function on dictionary it sorts in aplhabetical
order of value
print("="*50)
print(sortd_1)

# 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_1 = dict(my_counter_1)
sorted_subcat_dict_1 = dict(sorted(subcat_dict_1.items(), key=lambda kv:(kv[1] ,kv[0])))
```

```
{'appliedsciences': 2660, 'healthlifescience': 1183, 'specialneeds': 3324, 'literacy': 9272,
'earlydevelopment': 1294, 'mathematics': 7432, 'socialsciences': 598, 'historygeography': 887,
'esl': 1066, 'extracurricular': 158, 'visualarts': 1512, 'environmentalscience': 1443, 'literaturew
riting': 6458, 'gymfitness': 1792, 'music': 730, 'teamsports': 610, 'performingarts': 448,
'collegecareerprep': 575, 'other': 692, 'charactereducation': 663, 'foreignlanguages': 204, 'healt
hwellness': 4463, 'civicsgovernment': 243, 'economics': 51, 'communityservice': 123,
'financialliteracy': 62, 'nutritioneducation': 628, 'parentinvolvement': 94}
==================================================
[('appliedsciences', 2660), ('charactereducation', 663), ('civicsgovernment', 243),
('collegecareerprep', 575), ('communityservice', 123), ('earlydevelopment', 1294), ('economics',
51), ('environmentalscience', 1443), ('esl', 1066), ('extracurricular', 158),
('financialliteracy', 62), ('foreignlanguages', 204), ('gymfitness', 1792), ('healthlifescience',
1183), ('healthwellness', 4463), ('historygeography', 887), ('literacy', 9272),
('literaturewriting', 6458), ('mathematics', 7432), ('music', 730), ('nutritioneducation', 628), (
'other', 692), ('parentinvolvement', 94), ('performingarts', 448), ('socialsciences', 598),
('specialneeds', 3324), ('teamsports', 610), ('visualarts', 1512)]
```

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

Train_Data1

In [255]:

```
# merge two column text dataframe: https://stackoverflow.com/questions/19377969/combine-two-column
s-of-text-in-dataframe-in-pandas-python
train_data1["project_essay"] = train_data1["project_essay_1"].map(str) +train_data1["project_essay_
2"].map(str)+train_data1["project_essay_3"].map(str) +  train_data1["project_essay_4"].map(str)
            #Here the .map(str) converts string to all the coulmns in project_eassy_1/2/3/4
print(train_data1['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 [256]:

```
# printing some random essays.
print(train_data1['project_essay'].values[10])
print("="*50)
print(train_data1['project_essay'].values[20000])
print("="*50)
```

```
My students yearn for a classroom environment that matches their desire to learn. With education c
hanging daily, we need a classroom that can meet the needs of all of my first graders.I have the p
rivilege of teaching an incredible group of six and seven year olds who absolutely LOVE to learn.
I am completely blown away by their love for learning. Each day is a new adventure as they enjoy l
earning from nonfiction text and hands on activities. Many of my students are very active learners
who benefit from kinesthetic activities. Sometimes learning, while sitting in a seat, is
difficult. I want every child the opportunity to focus their energy in order to do their best in
school!Ideally, I would love to delve right into \"flexible seating\" where students are provided
many different seating options (chairs, hokki stools, on mats on the ground, etc.) and they have t
he freedom to choose which ever seat they feel they need. My student would be able to choose which
seating option will best help them learn. In addition, a pencil sharpener, mobile easel, magnetic
strips and mounting tape will help make our classroom better suited for 6 and 7 year olds.This pro
ject will be so beneficial for my students in that they will be able to better focus their energy.
Something so small, choosing their own seat, will help encourage a positive learning environment t
hat promotes learning for all students. The easel will help make our classroom more mobile, becaus
e it is both dry erase and on wheels. Magnetic strips, mounting tape and a pencil sharpener will a
```

```
flow for more resources for the students during the school day.
==================================================
\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the bi
ggest enthusiasm for learning. My students learn in many different ways using all of our senses an
d multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nSt
udents in my class come from a variety of different backgrounds which makes for wonderful sharing
of experiences and cultures, including Native Americans.\r\nOur school is a caring community of su
ccessful learners which can be seen through collaborative student project based learning in and ou
t of the classroom. Kindergarteners in my class love to work with hands-on materials and have many
different opportunities to practice a skill before it is mastered. Having the social skills to wor
k cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love to role play in our
pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try coo
king with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it's healthy for their bodies. This project w
ould expand our learning of nutrition and agricultural cooking recipes by having us peel our own a
pples to make homemade applesauce, make our own bread, and mix up healthy plants from our classroo
m garden in the spring. We will also create our own cookbooks to be printed and shared with famili
es. \r\nStudents will gain math and literature skills as well as a life long enjoyment for healthy
cooking.nannan
==================================================
```

In [257]:

```python
test1 = decontracted(train_data1['project_essay'].values[20000])
print(test1)
print("="*50)
```

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the smallest students with the b
iggest enthusiasm for learning. My students learn in many different ways using all of our senses a
nd multiple intelligences. I use a wide range of techniques to help all my students succeed. \r\nS
tudents in my class come from a variety of different backgrounds which makes for wonderful sharing
of experiences and cultures, including Native Americans.\r\nOur school is a caring community of su
ccessful learners which can be seen through collaborative student project based learning in and ou
t of the classroom. Kindergarteners in my class love to work with hands-on materials and have many
different opportunities to practice a skill before it is mastered. Having the social skills to wor
k cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love to role play in our
pretend kitchen in the early childhood classroom. I have had several kids ask me, \"Can we try coo
king with REAL food?\" I will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes by having us peel our own
apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classro
om garden in the spring. We will also create our own cookbooks to be printed and shared with famil
ies. \r\nStudents will gain math and literature skills as well as a life long enjoyment for health
y cooking.nannan
==================================================
```

In [258]:

```python
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
test1 = test1.replace('\\r', ' ')
test1 = test1.replace('\\"', ' ')
test1 = test1.replace('\\n', ' ')
print(test1)
```

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the smallest students with the big
gest enthusiasm for learning. My students learn in many different ways using all of our senses and
multiple intelligences. I use a wide range of techniques to help all my students succeed.
Students in my class come from a variety of different backgrounds which makes for wonderful
sharing of experiences and cultures, including Native Americans.  Our school is a caring community
of successful learners which can be seen through collaborative student project based learning in a
nd out of the classroom. Kindergarteners in my class love to work with hands-on materials and have
many different opportunities to practice a skill before it is mastered. Having the social skills t
o work cooperatively with friends is a crucial aspect of the kindergarten curriculum.Montana is
the perfect place to learn about agriculture and nutrition. My students love to role play in our p
retend kitchen in the early childhood classroom. I have had several kids ask me,  Can we try cooki
ng with REAL food?  I will take their idea and create  Common Core Cooking Lessons  where we learn
important math and writing concepts while cooking delicious healthy food for snack time. My
students will have a grounded appreciation for the work that went into making the food and knowled
```

```
ge of where the ingredients came from as well as how it is healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes by having us peel our own
apples to make homemade applesauce, make our own bread, and mix up healthy plants from our classro
om garden in the spring. We will also create our own cookbooks to be printed and shared with famil
ies.   Students will gain math and literature skills as well as a life long enjoyment for healthy
cooking.nannan
```

In [259]:

```
#remove special character: https://stackoverflow.com/a/5843547/4084039
test1 = re.sub('[^A-Za-z0-9]+', ' ', test1) #square bracket creates either or set; + signifes 1 or
more character
print(test1)
```

```
 A person is a person no matter how small Dr Seuss I teach the smallest students with the biggest
enthusiasm for learning My students learn in many different ways using all of our senses and multi
ple intelligences I use a wide range of techniques to help all my students succeed Students in my
class come from a variety of different backgrounds which makes for wonderful sharing of
experiences and cultures including Native Americans Our school is a caring community of successful
learners which can be seen through collaborative student project based learning in and out of the
classroom Kindergarteners in my class love to work with hands on materials and have many different
opportunities to practice a skill before it is mastered Having the social skills to work
cooperatively with friends is a crucial aspect of the kindergarten curriculum Montana is the
perfect place to learn about agriculture and nutrition My students love to role play in our
pretend kitchen in the early childhood classroom I have had several kids ask me Can we try cooking
with REAL food I will take their idea and create Common Core Cooking Lessons where we learn
important math and writing concepts while cooking delicious healthy food for snack time My
students will have a grounded appreciation for the work that went into making the food and knowled
ge of where the ingredients came from as well as how it is healthy for their bodies This project w
ould expand our learning of nutrition and agricultural cooking recipes by having us peel our own a
pples to make homemade applesauce make our own bread and mix up healthy plants from our classroom
garden in the spring We will also create our own cookbooks to be printed and shared with families
Students will gain math and literature skills as well as a life long enjoyment for healthy cooking
nannan
```

In [260]:

```
#Combining all the above statments to transform our text in a clean text
from tqdm import tqdm
preprocessed_essays1 = []
# tqdm is for printing the status bar
for sentance in tqdm(train_data1['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_essays1.append(sent.strip())
```

```
100%|████████████████████████████████████████████████████████████████████████████| 30000/30000
[00:04<00:00, 7058.81it/s]
```

In [261]:

```
#printing the text after preprocessing
preprocessed_essays1[0]
```

Out[261]:

```
'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
curriculum classroom although things like magnets classroom know use effectively kits provide righ
t amount materials show use appropriate way'
```

In [262]:

```
train_data1['preprocessed_essays']=preprocessed_essays1
train_data1.drop(['project_essay'], axis=1,inplace=True)
```

Project Title Text

In [263]:

```
from tqdm import tqdm
preprocessed_title1 = []
# tqdm is for printing the status bar
for title in tqdm(train_data1['project_title'].values):
    test_1 = decontracted(title)
    test_1 = test_1.replace('\\r', ' ')
    test_1 = test_1.replace('\\"', ' ')
    test_1 = test_1.replace('\\n', ' ')
    test_1 = re.sub('[^A-Za-z0-9]+', ' ', test_1)
    test_1=test_1.lower()
    # https://gist.github.com/sebleier/554280
    test_1 = ' '.join(e for e in test_1.split() if e not in s)
    preprocessed_title1.append(test_1.strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 30000/30000
[00:00<00:00, 72457.59it/s]
```

In [264]:

```
train_data1['preprocessed_title']=preprocessed_title1
train_data1.drop(['project_title'], axis=1,inplace=True)
```

Category Preprocessing

Teacher Prefix

In [266]:

```
from tqdm import tqdm
import string
preprocessed_prefix1=[]
for prefix in tqdm(train_data1['teacher_prefix'].values):
    test1=str(prefix).strip(".")
    test1=test1.lower()
    preprocessed_prefix1.append(test1)
```

```
100%|████████████████████████████████████████████████████████████████████████| 30000/30000
[00:00<00:00, 1361669.12it/s]
```

In [267]:

```
train_data1['preprocessed_prefix']=preprocessed_prefix1
train_data1.drop(['teacher_prefix'], axis=1,inplace=True)
```

Grade Category

In [268]:

```
preprocessed_grade1=[]
for grade in tqdm(train_data1['project_grade_category'].values):
    grade1=grade.strip(" ")
    grade1=grade1.replace(" ", "_")
    grade1=grade1.replace("-","_")
    preprocessed_grade1.append(grade1)
```

```
100%|██████████████████████████████████████████████████████████████████████| 30000/30000
[00:00<00:00, 1198520.96it/s]
```

In [269]:

```
train_data1['preprocessed_grade']=preprocessed_grade1
train_data1.drop(['project_grade_category'], axis=1,inplace=True)
```

Project Resource Summary

In [270]:

```
from tqdm import tqdm
preprocessed_resource1 = []
# tqdm is for printing the status bar
for resource in tqdm(train_data1['project_resource_summary'].values):
    sent1 = decontracted(resource)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    sent1=sent1.lower()
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e not in s)
    preprocessed_resource1.append(sent1.strip())
```

```
100%|████████████████████████████████████████████████████████████████████████| 30000/30000
[00:00<00:00, 39840.41it/s]
```

In [271]:

```
train_data1['preprocessed_resource']=preprocessed_resource1
train_data1.drop(['project_resource_summary'], axis=1,inplace=True)
```

Data Splitting

In [272]:

```
train_data1.columns
```

Out[272]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'school_state', 'Date',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'teacher_number_of_previously_posted_projects',
       'project_is_approved', 'price', 'quantity', 'clean_categories',
       'clean_subcategories', 'preprocessed_essays', 'preprocessed_title',
       'preprocessed_prefix', 'preprocessed_grade', 'preprocessed_resource'],
      dtype='object')
```

In [273]:

```
X1=train_data1.drop(columns=['id',"teacher_id","Date",'project_essay_1','project_essay_2','project_
essay_3','project_essay_4'])
```

In [274]:

```
print(X1.columns)
print("*"*50)
print(X1.head())
```

```
Index(['Unnamed: 0', 'school_state',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'preprocessed_essays', 'preprocessed_title', 'preprocessed_prefix',
       'preprocessed_grade', 'preprocessed_resource'],
      dtype='object')
**************************************************
   Unnamed: 0 school_state  teacher_number_of_previously_posted_projects  \
0        8393           CA                                            53   
1       37728           UT                                             4   
2       74477           CA                                            10   
3      100660           GA                                             2   
4       33679           WA                                             2   

   project_is_approved   price  quantity  clean_categories  \
0                    1  725.05         4       mathscience   
1                    1  213.03         8      specialneeds   
2                    1  329.00         1  literacylanguage   
3                    1  481.04         9   appliedlearning   
4                    1   17.74        14  literacylanguage   

                 clean_subcategories  \
0  appliedsciences healthlifescience   
1                       specialneeds   
2                           literacy   
3                   earlydevelopment   
4                           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...   
3  recently read article giving students choice l...   
4  students crave challenge eat obstacles breakfa...   

                        preprocessed_title preprocessed_prefix  \
0      engineering steam primary classroom                 mrs   
1                      sensory tools focus                  ms   
2  mobile learning mobile listening center                 mrs   
3       flexible seating flexible learning                 mrs   
4            going deep art inner thinking                 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...  
3      Grades_PreK_2  students need flexible seating classroom choos...  
4         Grades_3_5  students need copies new york times best selle...  
```

In [275]:

```
y1=X1['project_is_approved']
```

In [276]:

```
X1=X1.drop(columns=['project_is_approved'])
```

In [277]:

```
print(X1.shape)
print("="*50)
print(y1.shape)
```

```
(30000, 12)
==================================================
(30000,)
```

In [278]:

```
X1.columns
```

Out[278]:

```
Index(['Unnamed: 0', 'school_state',
        'teacher_number_of_previously_posted_projects', 'price', 'quantity',
        'clean_categories', 'clean_subcategories', 'preprocessed_essays',
        'preprocessed_title', 'preprocessed_prefix', 'preprocessed_grade',
        'preprocessed_resource'],
       dtype='object')
```

In [279]:

```
# 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_11, X_test_1, y_11, y_test_1 =model_selection.train_test_split(X1,y1, test_size=0.33, random_stat
e=4)#random spliiting of data into test and train
```

In [280]:

```
X_train_1, X_cv_1, y_train_1, y_cv_1 = train_test_split(X_11, y_11, test_size=0.33,random_state=4)
# this is random splitting of train data into train anc cross-validation
```

In [281]:

```
print(X_train_1.shape, y_train_1.shape)
print(X_cv_1.shape, y_cv_1.shape)
print(X_test_1.shape, y_test_1.shape)

print("="*100)
```

```
(13467, 12) (13467,)
(6633, 12) (6633,)
(9900, 12) (9900,)
====================================================================================================
```

Vectorization

Response Encoding of categorical feature

Category Feature

In [284]:

```
init_data=pd.DataFrame(columns=['categories','label'])
```

In [285]:

```
init_data['categories']=X_train_1['clean_categories']
init_data['label']=y_train_1
```

In [286]:

```
print(init_data.head())
print(init_data.shape)
```

```
                         categories  label
28326                 historycivics      1
23885                  healthsports      1
8742               literacylanguage      0
18625  appliedlearning specialneeds      1
27632                  healthsports      1
(13467, 2)
```

In [288]:

```
#how to calculate conditional probability python pandas -
>https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
```

```
#https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
n-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
cond_prob.shape
```

Out[288]:

```
(42,)
```

In [289]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div
(cond_prob , axis=0, level='categories'),columns=['probability'])
```

In [290]:

```
print(encoded_cat.head())
print(encoded_cat.tail())
```

```
                                        probability
label categories                                   
0     appliedlearning                      0.191142
      appliedlearning healthsports         0.131579
      appliedlearning historycivics        0.222222
      appliedlearning literacylanguage     0.156997
      appliedlearning mathscience          0.165289
                                 probability
label categories                            
1     musicarts historycivics       1.000000
      musicarts specialneeds        1.000000
      specialneeds                  0.824719
      specialneeds healthsports     0.750000
      specialneeds musicarts        0.864865
```

In [291]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[291]:

```
(79, 3)
```

In [292]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
```

In [293]:

```
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label                        categories  probability
0      0                   appliedlearning     0.191142
1      0      appliedlearning healthsports     0.131579
2      0     appliedlearning historycivics     0.222222
3      0  appliedlearning literacylanguage     0.156997
4      0       appliedlearning mathscience     0.165289
(38, 3)
    label                        categories  probability
38      1                   appliedlearning     0.808858
39      1      appliedlearning healthsports     0.868421
40      1     appliedlearning historycivics     0.777778
41      1  appliedlearning literacylanguage     0.843003
42      1       appliedlearning mathscience     0.834711
(41, 3)
```

In [294]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [295]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  0
idx= 0
idx = 1
temp1=  1
idx= 1
idx = 2
temp1=  2
idx= 2
idx = 3
temp1=  3
idx= 3
idx = 4
temp1=  4
idx= 4
idx = 5
temp1=  5
idx= 5
idx = 6
temp1=  6
idx= 6
idx = 7
temp1=  7
idx= 7
idx = 8
temp1=  8
idx= 8
idx = 9
temp1=  9
idx= 9
idx = 10
temp1=  10
idx= 10
idx = 11
temp1=  11
idx= 11
idx = 12
temp1=  12
idx= 12
idx = 13
temp1=  13
idx= 13
idx = 14
temp1=  14
```

```
idx= 14
idx = 15
temp1=  17
idx= 15
idx = 16
temp1=  18
idx= 16
idx = 17
temp1=  19
idx= 17
idx = 18
temp1=  20
idx= 18
idx = 19
temp1=  21
idx= 19
idx = 20
temp1=  22
idx= 20
idx = 21
temp1=  23
idx= 21
idx = 22
temp1=  24
idx= 22
idx = 23
temp1=  25
idx= 23
idx = 24
temp1=  26
idx= 24
idx = 25
temp1=  27
idx= 25
idx = 26
temp1=  28
idx= 26
idx = 27
temp1=  29
idx= 27
idx = 28
temp1=  30
idx= 28
idx = 29
temp1=  31
idx= 29
idx = 30
temp1=  32
idx= 30
idx = 31
temp1=  33
idx= 31
idx = 32
temp1=  34
idx= 32
idx = 33
temp1=  35
idx= 33
idx = 34
idx = 35
temp1=  38
idx= 35
idx = 36
temp1=  39
idx= 36
idx = 37
temp1=  40
idx= 37
idx = 38
idx = 39
idx = 40
```

In [296]:

```
c_0=[]
c_1=[]
```

```
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])

print(len(c_0))
print(len(c_1))
```

```
41
41
```

In [297]:

```
a=X_train_1['clean_categories'].unique()
a=a[0:41]
a.shape
```

Out[297]:

```
(41,)
```

In [298]:

```
#Creating A Response Table
res_table=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table['prob_0']=c_0
res_table['prob_1']=c_1
#res_table['label']=label
res_table['categories']=a
```

In [299]:

```
res_table.shape
```

Out[299]:

```
(41, 3)
```

Training based on response_table

Train Data

In [300]:

```
train_coded_cat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [301]:

```
temp_0=[]
temp_1=[]
for cat in X_train_1["clean_categories"].values:
    if cat in res_table["categories"].values:
        temp_0.append(res_table.loc[cat,"prob_0"])
        temp_1.append(res_table.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [302]:

```
train_coded_cat["prob_0"]=temp_0
train_coded_cat["prob_1"]=temp_1
```

In [303]:

```
train_coded_cat.shape
train_coded_cat.shape
```

Out[303]:

```
(13467, 2)
```

CV Data

In [304]:

```
cv_coded_cat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [305]:

```
temp_0=[]
temp_1=[]
for cat in X_cv_1["clean_categories"].values:
    if cat in res_table["categories"].values:
        temp_0.append(res_table.loc[cat,"prob_0"])
        temp_1.append(res_table.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [306]:

```
cv_coded_cat["prob_0"]=temp_0
cv_coded_cat["prob_1"]=temp_1
```

In [307]:

```
cv_coded_cat.shape
```

Out[307]:

```
(6633, 2)
```

Test Data

In [308]:

```
test_coded_cat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [309]:

```
temp_0=[]
temp_1=[]
for cat in X_test_1["clean_categories"].values:
    if cat in res_table["categories"].values:
        temp_0.append(res_table.loc[cat,"prob_0"])
        temp_1.append(res_table.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [310]:

```
test_coded_cat["prob_0"]=temp_0
test_coded_cat["prob_1"]=temp_1
```

In [311]:

```
test_coded_cat.shape
```

Out[311]:

```
([illegible], 2)
```

```
(9900, 2)
```

Sub_category

In [312]:

```
init_data=pd.DataFrame(columns=['categories','label'])
init_data['categories']=X_train_1['clean_subcategories']
init_data['label']=y_train_1
```

In [313]:

```
print(init_data.head())
print(init_data.shape)
```

```
                           categories  label
28326      economics financialliteracy      1
23885               nutritioneducation      1
8742                          literacy      0
18625  charactereducation specialneeds      1
27632                   healthwellness      1
(13467, 2)
```

In [314]:

```
#how to calculate conditional probability python pandas -
>https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
n-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
```

In [315]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div
(cond_prob , axis=0, level='categories'),columns=['probability'])
```

In [316]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[316]:

```
(472, 3)
```

In [317]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
```

In [318]:

```
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label                          categories  probability
0      0                     appliedsciences     0.213115
1      0  appliedsciences charactereducation     0.250000
2      0   appliedsciences collegecareerprep     0.222222
3      0    appliedsciences communityservice     1.000000
4      0    appliedsciences earlydevelopment     0.121212
(190, 3)
     label                          categories  probability
190      1                     appliedsciences     0.786885
191      1  appliedsciences charactereducation     0.750000
192      1    appliedsciences civicsgovernment     1.000000
```

```
193      1    appliedsciences collegecareerprep     0.777778
194      1    appliedsciences earlydevelopment     0.878788
(282, 3)
```

In [319]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [320]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  0
idx= 0
idx = 1
temp1=  1
idx= 1
idx = 2
temp1=  3
idx= 2
idx = 3
idx = 4
temp1=  4
idx= 4
idx = 5
temp1=  5
idx= 5
idx = 6
temp1=  6
idx= 6
idx = 7
temp1=  7
idx= 7
idx = 8
temp1=  8
idx= 8
idx = 9
temp1=  10
idx= 9
idx = 10
temp1=  12
idx= 10
idx = 11
temp1=  13
idx= 11
idx = 12
temp1=  14
idx= 12
idx = 13
temp1=  15
idx= 13
idx = 14
idx = 14
temp1=  18
idx= 14
idx = 15
temp1=  20
idx= 15
idx = 16
temp1=  21
idx= 16
idx = 17
temp1=  23
idx= 17
idx = 18
temp1=  24
idx= 18
idx = 19
temp1=  26
idx= 19
idx = 20
temp1=  28
idx= 20
idx = 21
temp1=  30
idx= 21
idx = 22
temp1=  31
idx= 22
idx = 23
idx = 24
temp1=  33
idx= 24
idx = 25
idx = 26
temp1=  34
idx= 26
idx = 27
temp1=  35
idx= 27
idx = 28
temp1=  36
idx= 28
idx = 29
temp1=  37
idx= 29
idx = 30
temp1=  38
idx= 30
idx = 31
temp1=  39
idx= 31
idx = 32
temp1=  42
idx= 32
idx = 33
temp1=  43
idx= 33
idx = 34
temp1=  45
idx= 34
idx = 35
temp1=  48
idx= 35
idx = 36
temp1=  49
idx= 36
idx = 37
temp1=  53
idx= 37
idx = 38
temp1=  54
idx= 38
idx = 39
temp1=  55
idx= 39
idx = 40
temp1=  59
idx= 40
idx = 41
idx = 41
temp1=  60
idx= 41
idx = 42
temp1=  61
idx= 42
idx = 43
temp1=  65
idx= 43
idx = 44
temp1=  70
idx= 44
idx = 45
temp1=  71
idx= 45
idx = 46
temp1=  72
idx= 46
idx = 47
idx = 48
temp1=  73
idx= 48
idx = 49
temp1=  75
idx= 49
idx = 50
idx = 51
temp1=  76
idx= 51
idx = 52
temp1=  77
idx= 52
idx = 53
temp1=  78
idx= 53
idx = 54
temp1=  81
idx= 54
idx = 55
temp1=  89
idx= 55
idx = 56
temp1=  92
idx= 56
idx = 57
temp1=  93
idx= 57
idx = 58
temp1=  94
idx= 58
idx = 59
temp1=  95
idx= 59
idx = 60
temp1=  98
idx= 60
idx = 61
temp1=  99
idx= 61
idx = 62
temp1=  100
idx= 62
idx = 63
temp1=  101
idx= 63
idx = 64
temp1=  102
idx= 64
idx = 65
temp1=  103
idx= 65
idx = 66
temp1=  104
idx= 66
idx = 67
idx = 68
temp1=  107
idx= 68
```

```
idx= 68
idx = 69
temp1=  109
idx= 69
idx = 70
temp1=  110
idx= 70
idx = 71
temp1=  117
idx= 71
idx = 72
temp1=  118
idx= 72
idx = 73
temp1=  121
idx= 73
idx = 74
temp1=  123
idx= 74
idx = 75
temp1=  124
idx= 75
idx = 76
temp1=  125
idx= 76
idx = 77
temp1=  126
idx= 77
idx = 78
temp1=  127
idx= 78
idx = 79
temp1=  129
idx= 79
idx = 80
temp1=  130
idx= 80
idx = 81
temp1=  131
idx= 81
idx = 82
temp1=  132
idx= 82
idx = 83
temp1=  133
idx= 83
idx = 84
temp1=  134
idx= 84
idx = 85
idx = 86
temp1=  135
idx= 86
idx = 87
temp1=  136
idx= 87
idx = 88
idx = 89
temp1=  137
idx= 89
idx = 90
temp1=  138
idx= 90
idx = 91
temp1=  139
idx= 91
idx = 92
temp1=  140
idx= 92
idx = 93
temp1=  142
idx= 93
idx = 94
temp1=  143
idx= 94
idx = 95
temp1=  144
idx= 95
```

```
idx= 95
idx = 96
temp1=  145
idx= 96
idx = 97
temp1=  146
idx= 97
idx = 98
temp1=  147
idx= 98
idx = 99
temp1=  148
idx= 99
idx = 100
temp1=  149
idx= 100
idx = 101
temp1=  157
idx= 101
idx = 102
temp1=  160
idx= 102
idx = 103
temp1=  161
idx= 103
idx = 104
temp1=  162
idx= 104
idx = 105
temp1=  165
idx= 105
idx = 106
temp1=  166
idx= 106
idx = 107
temp1=  167
idx= 107
idx = 108
temp1=  172
idx= 108
idx = 109
temp1=  174
idx= 109
idx = 110
temp1=  176
idx= 110
idx = 111
temp1=  178
idx= 111
idx = 112
temp1=  180
idx= 112
idx = 113
temp1=  183
idx= 113
idx = 114
temp1=  184
idx= 114
idx = 115
temp1=  185
idx= 115
idx = 116
temp1=  186
idx= 116
idx = 117
temp1=  187
idx= 117
idx = 118
temp1=  188
idx= 118
idx = 119
temp1=  189
idx= 119
idx = 120
temp1=  190
idx= 120
idx = 121
temp1=  192
```

```
temp1=  192
idx= 121
idx = 122
temp1=  193
idx= 122
idx = 123
temp1=  194
idx= 123
idx = 124
temp1=  195
idx= 124
idx = 125
temp1=  197
idx= 125
idx = 126
temp1=  198
idx= 126
idx = 127
temp1=  199
idx= 127
idx = 128
temp1=  200
idx= 128
idx = 129
temp1=  201
idx= 129
idx = 130
temp1=  202
idx= 130
idx = 131
temp1=  203
idx= 131
idx = 132
temp1=  204
idx= 132
idx = 133
temp1=  205
idx= 133
idx = 134
idx = 135
temp1=  208
idx= 135
idx = 136
temp1=  209
idx= 136
idx = 137
temp1=  210
idx= 137
idx = 138
temp1=  211
idx= 138
idx = 139
temp1=  212
idx= 139
idx = 140
temp1=  213
idx= 140
idx = 141
temp1=  214
idx= 141
idx = 142
temp1=  218
idx= 142
idx = 143
temp1=  219
idx= 143
idx = 144
temp1=  220
idx= 144
idx = 145
temp1=  221
idx= 145
idx = 146
temp1=  222
idx= 146
idx = 147
temp1=  223
idx= 147
```

```
idx= 147
idx = 148
temp1=  226
idx= 148
idx = 149
temp1=  227
idx= 149
idx = 150
temp1=  228
idx= 150
idx = 151
temp1=  229
idx= 151
idx = 152
temp1=  230
idx= 152
idx = 153
temp1=  231
idx= 153
idx = 154
temp1=  232
idx= 154
idx = 155
temp1=  233
idx= 155
idx = 156
temp1=  234
idx= 156
idx = 157
temp1=  236
idx= 157
idx = 158
temp1=  237
idx= 158
idx = 159
temp1=  238
idx= 159
idx = 160
temp1=  239
idx= 160
idx = 161
temp1=  240
idx= 161
idx = 162
temp1=  241
idx= 162
idx = 163
temp1=  242
idx= 163
idx = 164
temp1=  245
idx= 164
idx = 165
temp1=  248
idx= 165
idx = 166
temp1=  249
idx= 166
idx = 167
temp1=  251
idx= 167
idx = 168
temp1=  252
idx= 168
idx = 169
temp1=  253
idx= 169
idx = 170
temp1=  257
idx= 170
idx = 171
temp1=  260
idx= 171
idx = 172
temp1=  262
idx= 172
idx = 173
idx = 174
idx = 174
temp1=  264
idx= 174
idx = 175
temp1=  265
idx= 175
idx = 176
temp1=  266
idx= 176
idx = 177
temp1=  268
idx= 177
idx = 178
temp1=  269
idx= 178
idx = 179
temp1=  270
idx= 179
idx = 180
idx = 181
temp1=  273
idx= 181
idx = 182
temp1=  274
idx= 182
idx = 183
temp1=  275
idx= 183
idx = 184
temp1=  276
idx= 184
idx = 185
temp1=  277
idx= 185
idx = 186
temp1=  278
idx= 186
idx = 187
temp1=  279
idx= 187
idx = 188
temp1=  280
idx= 188
idx = 189
temp1=  281
idx= 189
idx = 190
idx = 191
idx = 192
idx = 193
idx = 194
idx = 195
idx = 196
idx = 197
idx = 198
idx = 199
idx = 200
idx = 201
idx = 202
idx = 203
idx = 204
idx = 205
idx = 206
idx = 207
idx = 208
idx = 209
idx = 210
idx = 211
idx = 212
idx = 213
idx = 214
idx = 215
idx = 216
idx = 217
idx = 218
idx = 219
idx = 220
idx = 221
idx = 222
idx = 223
idx = 224
idx = 225
idx = 226
idx = 227
idx = 228
idx = 229
idx = 230
idx = 231
idx = 232
idx = 233
idx = 234
idx = 235
idx = 236
idx = 237
idx = 238
idx = 239
idx = 240
idx = 241
idx = 242
idx = 243
idx = 244
idx = 245
idx = 246
idx = 247
idx = 248
idx = 249
idx = 250
idx = 251
idx = 252
idx = 253
idx = 254
idx = 255
idx = 256
idx = 257
idx = 258
idx = 259
idx = 260
idx = 261
idx = 262
idx = 263
idx = 264
idx = 265
idx = 266
idx = 267
idx = 268
idx = 269
idx = 270
idx = 271
idx = 272
idx = 273
idx = 274
idx = 275
idx = 276
idx = 277
idx = 278
idx = 279
idx = 280
idx = 281
```

In [321]:

```
c_0=[]
c_1=[]
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])


print(len(c_0))
print(len(c_1))
print(len(label))
```

```
282
282
282
```

In [323]:

```
a=X_train_1['clean_subcategories'].unique()
a=a[0:282]
len(a)
```

Out[323]:

```
282
```

In [324]:

```
#Creating A Response Table
res_table_subcat=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table_subcat['prob_0']=c_0
res_table_subcat['prob_1']=c_1
#res_table['label']=label
res_table_subcat['categories']=a
```

In [325]:

```
res_table_subcat.head()
```

Out[325]:

	prob_0	prob_1	categories
economics financialliteracy	0.213115	0.786885	economics financialliteracy
nutritioneducation	0.250000	0.750000	nutritioneducation
literacy	0.222222	0.777778	literacy
charactereducation specialneeds	0.000000	0.777778	charactereducation specialneeds
healthwellness	0.121212	0.878788	healthwellness

Training based on response_table

Train Data

In [326]:

```
train_coded_subcat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [327]:

```
temp_0=[]
temp_1=[]
for cat in X_train_1["clean_subcategories"].values:
    if cat in res_table_subcat["categories"].values:
        temp_0.append(res_table_subcat.loc[cat,"prob_0"])
        temp_1.append(res_table_subcat.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [328]:

```
train_coded_subcat["prob_0"]=temp_0
train_coded_subcat["prob_1"]=temp_1
```

In [329]:

```
train_coded_subcat.shape
```

Out[329]:

```
(13467, 2)
```

CV data

In [330]:

```
cv_coded_subcat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [331]:

```
temp_0=[]
temp_1=[]
for cat in X_cv_1["clean_subcategories"].values:
    if cat in res_table_subcat["categories"].values:
        temp_0.append(res_table_subcat.loc[cat,"prob_0"])
        temp_1.append(res_table_subcat.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [332]:

```
cv_coded_subcat["prob_0"]=temp_0
cv_coded_subcat["prob_1"]=temp_1
```

In [333]:

```
cv_coded_subcat.shape
```

Out[333]:

```
(6633, 2)
```

Test data

In [334]:

```
test_coded_subcat=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [335]:

```
temp_0=[]
temp_1=[]
for cat in X_test_1["clean_subcategories"].values:
    if cat in res_table_subcat["categories"].values:
        temp_0.append(res_table_subcat.loc[cat,"prob_0"])
        temp_1.append(res_table_subcat.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [336]:

```
test_coded_subcat["prob_0"]=temp_0
test_coded_subcat["prob_1"]=temp_1
```

In [337]:

```
test_coded_subcat.shape
```

Out[337]:

```
(9900, 2)
```

Teacher_Prefix

In [338]:

```
init_data=pd.DataFrame(columns=['categories','label'])
init_data['categories']=X_train_1['preprocessed_prefix']
init_data['label']=y_train_1
```

In [339]:

```
print(init_data.head())
print(init_data.shape)
```

```
      categories  label
28326        mrs      1
23885         ms      1
8742         mrs      0
18625         ms      1
27632        mrs      1
(13467, 2)
```

In [340]:

```
#how to calculate conditional probability python pandas -
>https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
n-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
```

In [341]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div
(cond_prob , axis=0, level='categories'),columns=['probability'])
```

In [342]:

```
print(encoded_cat.head())
print(encoded_cat.tail())
```

```
                  probability
label categories             
0     mr             0.139878
      mrs            0.149141
      ms             0.161505
      teacher        0.210909
1     dr             1.000000
                  probability
label categories             
1     dr             1.000000
      mr             0.860122
      mrs            0.850859
      ms             0.838495
      teacher        0.789091
```

In [343]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[343]:

```
(9, 3)
```

In [344]:

In [344]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label categories  probability
0      0         mr     0.139878
1      0        mrs     0.149141
2      0         ms     0.161505
3      0    teacher     0.210909
(4, 3)
   label categories  probability
4      1         dr     1.000000
5      1         mr     0.860122
6      1        mrs     0.850859
7      1         ms     0.838495
8      1    teacher     0.789091
(5, 3)
```

In [345]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [346]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  1
idx= 0
idx = 1
temp1=  2
idx= 1
idx = 2
temp1=  3
idx= 2
idx = 3
temp1=  4
idx= 3
idx = 4
```

In [347]:

```
c_0=[]
c_1=[]
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])


print(len(c_0))
print(len(c_1))
print(len(label))
```

```
5
5
5
```

In [350]:

```
a=X_train_1['preprocessed_prefix'].unique()
len(a)
```

Out[350]:

```
5
```

In [351]:

```
#Creating A Response Table
res_table_prefix=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table_prefix['prob_0']=c_0
res_table_prefix['prob_1']=c_1
#res_table['label']=label
res_table_prefix['categories']=a
```

In [352]:

```
res_table_prefix.head()
```

Out[352]:

	prob_0	prob_1	categories
mrs	0.139878	0.860122	mrs
ms	0.149141	0.850859	ms
mr	0.161505	0.838495	mr
teacher	0.210909	0.789091	teacher
dr	0.000000	0.789091	dr

Training Based on Response Table

Train Data

In [353]:

```
train_coded_prefix=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [354]:

```
temp_0=[]
temp_1=[]
for cat in X_train_1["preprocessed_prefix"].values:
    if cat in res_table_prefix["categories"].values:
        temp_0.append(res_table_prefix.loc[cat,"prob_0"])
        temp_1.append(res_table_prefix.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [355]:

```
train_coded_prefix["prob_0"]=temp_0
train_coded_prefix["prob_1"]=temp_1
```

In [356]:

```
train_coded_prefix.shape
```

Out[356]:

```
(13467, 2)
```

CV data

In [357]:

```
cv_coded_prefix=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [358]:

```
temp_0=[]
temp_1=[]
for cat in X_cv_1["preprocessed_prefix"].values:
    if cat in res_table_prefix["categories"].values:
        temp_0.append(res_table_prefix.loc[cat,"prob_0"])
        temp_1.append(res_table_prefix.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [359]:

```
cv_coded_prefix["prob_0"]=temp_0
cv_coded_prefix["prob_1"]=temp_1
```

In [360]:

```
cv_coded_prefix.shape
```

Out[360]:

```
(6633, 2)
```

Test Data

In [361]:

```
test_coded_prefix=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [362]:

```
temp_0=[]
temp_1=[]
for cat in X_test_1["preprocessed_prefix"].values:
    if cat in res_table_prefix["categories"].values:
        temp_0.append(res_table_prefix.loc[cat,"prob_0"])
        temp_1.append(res_table_prefix.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [363]:

```
test_coded_prefix["prob_0"]=temp_0
test_coded_prefix["prob_1"]=temp_1
```

In [364]:

```
test_coded_prefix.shape
```

Out[364]:

```
(9900, 2)
```

Grade Category

In [373]:

```
init_data=pd.DataFrame(columns=['categories','label'])


init_data['categories']=X_train_1['preprocessed_grade']
init_data['label']=y_train_1
```

In [374]:

```
print(init_data.head())
print(init_data.shape)
```

```
          categories  label
28326  Grades_PreK_2      1
23885     Grades_3_5      1
8742      Grades_3_5      0
18625     Grades_3_5      1
27632  Grades_PreK_2      1
(13467, 2)
```

In [375]:

```
#how to calculate conditional probability python pandas -
>https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
n-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
```

In [376]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div
(cond_prob , axis=0, level='categories'),columns=['probability'])
```

In [377]:

```
print(encoded_cat.head())
print(encoded_cat.tail())
```

```
                     probability
label categories
0     Grades_3_5        0.153863
      Grades_6_8        0.157315
      Grades_9_12       0.150992
      Grades_PreK_2     0.153288
1     Grades_3_5        0.846137
                     probability
label categories
0     Grades_PreK_2     0.153288
1     Grades_3_5        0.846137
      Grades_6_8        0.842685
      Grades_9_12       0.849008
      Grades_PreK_2     0.846712
```

In [378]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[378]:

```
(8, 3)
```

In [379]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label     categories  probability
0      0     Grades_3_5     0.153863
1      0     Grades_6_8     0.157315
2      0    Grades_9_12     0.150992
3      0  Grades_PreK_2     0.153288
(4, 3)
   label     categories  probability
4      1     Grades_3_5     0.846137
5      1     Grades_6_8     0.842685
6      1    Grades_9_12     0.849008
7      1  Grades_PreK_2     0.846712
(4, 3)
```

In [380]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [381]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  0
idx= 0
idx = 1
temp1=  1
idx= 1
idx = 2
temp1=  2
idx= 2
idx = 3
temp1=  3
idx= 3
```

In [382]:

```
c_0=[]
c_1=[]
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])


print(len(c_0))
print(len(c_1))
print(len(label))
```

```
4
4
4
```

In [383]:

```
a=X_train_1['preprocessed_grade'].unique()
len(a)
```

Out[383]:

```
4
```

In [384]:

```
#Creating A Response Table
res_table_grade=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table_grade['prob_0']=c_0
res_table_grade['prob_1']=c_1
#res_table['label']=label
res_table_grade['categories']=a
```

In [385]:

```
res_table_grade.head()
```

Out[385]:

	prob_0	prob_1	categories
Grades_PreK_2	0.153863	0.846137	Grades_PreK_2
Grades_3_5	0.157315	0.842685	Grades_3_5
Grades_6_8	0.150992	0.849008	Grades_6_8
Grades_9_12	0.153288	0.846712	Grades_9_12

Training Based on Response Table

Train Data

In [390]:

```
train_coded_grade=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [388]:

```
temp_0=[]
temp_1=[]
for cat in X_train_1['preprocessed_grade'].values:
    if cat in res_table_grade["categories"].values:
        temp_0.append(res_table_grade.loc[cat,"prob_0"])
        temp_1.append(res_table_grade.loc[cat,"prob_1"])
    else:
```

```
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [391]:

```
train_coded_grade["prob_0"]=temp_0
train_coded_grade["prob_1"]=temp_1
```

In [392]:

```
train_coded_grade.shape
```

Out[392]:

```
(13467, 2)
```

CV data

In [393]:

```
cv_coded_grade=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [394]:

```
temp_0=[]
temp_1=[]
for cat in X_cv_1['preprocessed_grade'].values:
    if cat in res_table_grade["categories"].values:
        temp_0.append(res_table_grade.loc[cat,"prob_0"])
        temp_1.append(res_table_grade.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [395]:

```
cv_coded_grade["prob_0"]=temp_0
cv_coded_grade["prob_1"]=temp_1
```

In [396]:

```
cv_coded_grade.shape
```

Out[396]:

```
(6633, 2)
```

Test Data

In [397]:

```
test_coded_grade=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [398]:

```
temp_0=[]
temp_1=[]
for cat in X_test_1['preprocessed_grade'].values:
    if cat in res_table_grade["categories"].values:
        temp_0.append(res_table_grade.loc[cat,"prob_0"])
        temp_1.append(res_table_grade.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [399]:

```
test_coded_grade["prob_0"]=temp_0
test_coded_grade["prob_1"]=temp_1
```

In [400]:

```
test_coded_grade.shape
```

Out[400]:

```
(9900, 2)
```

School_State Feature

In [401]:

```
init_data=pd.DataFrame(columns=['categories','label'])


init_data['categories']=X_train_1['school_state']
init_data['label']=y_train_1



print(init_data.head())
print(init_data.shape)
```

```
      categories  label
28326         AR      1
23885         FL      1
8742          MO      0
18625         NC      1
27632         IN      1
(13467, 2)
```

In [402]:

```
#how to calculate conditional probability python pandas -
>https://stackoverflow.com/questions/37818063/how-to-calculate-conditional-probability-of-values-i
n-dataframe-pandas-python
cond_prob=init_data.groupby('categories').size().div(len(init_data))
```

In [403]:

```
encoded_cat=pd.DataFrame(init_data.groupby(['label', 'categories']).size().div(len(init_data)).div
(cond_prob , axis=0, level='categories'),columns=['probability'])



print(encoded_cat.head())
print(encoded_cat.tail())
```

```
                  probability
label categories
0     AK             0.068966
      AL             0.193750
      AR             0.154472
      AZ             0.154930
      CA             0.140181
                  probability
label categories
1     VT             0.800000
      WA             0.885993
      WI             0.869792
      WV             0.772727
      WY             0.714286
```

In [404]:

```
encoded_cat.reset_index(inplace= True)
encoded_cat.shape
```

Out[404]:

```
(102, 3)
```

In [405]:

```
cat_1=encoded_cat[encoded_cat['label']==1]
cat_0=encoded_cat[encoded_cat['label']==0]
print(cat_0.head())
print(cat_0.shape)
print(cat_1.head())
print(cat_1.shape)
```

```
   label categories  probability
0      0         AK     0.068966
1      0         AL     0.193750
2      0         AR     0.154472
3      0         AZ     0.154930
4      0         CA     0.140181
(51, 3)
    label categories  probability
51      1         AK     0.931034
52      1         AL     0.806250
53      1         AR     0.845528
54      1         AZ     0.845070
55      1         CA     0.859819
(51, 3)
```

In [406]:

```
cat_1=cat_1.reset_index().drop(['index'], axis=1)
cat_0=cat_0.reset_index().drop(['index'], axis=1)
```

In [407]:

```
#Now making a response table
encoding_cat=[]
for idx in range(len(cat_1)):
    print("idx =", idx)
    try:
        temp1=cat_1.loc[cat_1['categories']==cat_0.iloc[idx]['categories']].index[0]
        print("temp1= ", temp1)
        if cat_0.iloc[idx]['categories'] in cat_1.iloc[temp1]['categories']:
            print("idx=" , idx)
            if(cat_0.iloc[idx]['probability'] > cat_1.iloc[temp1]['probability']):
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],0])
            else :
                encoding_cat.append([cat_0.iloc[idx]['probability'],cat_1.iloc[temp1]['probability'
],1])
        else:
            encoding_cat.append([cat_0.iloc[idx]['probability'],0,0])
            continue
            if cat_1.iloc[idx]['categories'] in cat_0.iloc[idx]['categories'] :
                encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
    except :
        encoding_cat.append([0,cat_1.iloc[idx]['probability'],1])
```

```
idx = 0
temp1=  0
idx= 0
idx = 1
temp1=  1
idx= 1
idx = 2
temp1=  2
idx= 2
idx = 3
```

```
temp1=  3
idx= 3
idx = 4
temp1=  4
idx= 4
idx = 5
temp1=  5
idx= 5
idx = 6
temp1=  6
idx= 6
idx = 7
temp1=  7
idx= 7
idx = 8
temp1=  8
idx= 8
idx = 9
temp1=  9
idx= 9
idx = 10
temp1=  10
idx= 10
idx = 11
temp1=  11
idx= 11
idx = 12
temp1=  12
idx= 12
idx = 13
temp1=  13
idx= 13
idx = 14
temp1=  14
idx= 14
idx = 15
temp1=  15
idx= 15
idx = 16
temp1=  16
idx= 16
idx = 17
temp1=  17
idx= 17
idx = 18
temp1=  18
idx= 18
idx = 19
temp1=  19
idx= 19
idx = 20
temp1=  20
idx= 20
idx = 21
temp1=  21
idx= 21
idx = 22
temp1=  22
idx= 22
idx = 23
temp1=  23
idx= 23
idx = 24
temp1=  24
idx= 24
idx = 25
temp1=  25
idx= 25
idx = 26
temp1=  26
idx= 26
idx = 27
temp1=  27
idx= 27
idx = 28
temp1=  28
idx= 28
```

```
idx = 29
temp1=  29
idx= 29
idx = 30
temp1=  30
idx= 30
idx = 31
temp1=  31
idx= 31
idx = 32
temp1=  32
idx= 32
idx = 33
temp1=  33
idx= 33
idx = 34
temp1=  34
idx= 34
idx = 35
temp1=  35
idx= 35
idx = 36
temp1=  36
idx= 36
idx = 37
temp1=  37
idx= 37
idx = 38
temp1=  38
idx= 38
idx = 39
temp1=  39
idx= 39
idx = 40
temp1=  40
idx= 40
idx = 41
temp1=  41
idx= 41
idx = 42
temp1=  42
idx= 42
idx = 43
temp1=  43
idx= 43
idx = 44
temp1=  44
idx= 44
idx = 45
temp1=  45
idx= 45
idx = 46
temp1=  46
idx= 46
idx = 47
temp1=  47
idx= 47
idx = 48
temp1=  48
idx= 48
idx = 49
temp1=  49
idx= 49
idx = 50
temp1=  50
idx= 50
```

In [408]:

```
c_0=[]
c_1=[]
label=[]
for i in encoding_cat:
    c_0.append(i[0])
    c_1.append(i[1])
    label.append(i[2])
```

```
print(len(c_0))
print(len(c_1))
print(len(label))
```

```
51
51
51
```

In [409]:

```
a=X_train_1['school_state'].unique()
len(a)
```

Out[409]:

```
51
```

In [410]:

```
#Creating A Response Table
res_table_state=pd.DataFrame(columns=['prob_0','prob_1','categories'], index=a)
res_table_state['prob_0']=c_0
res_table_state['prob_1']=c_1
#res_table['label']=label
res_table_state['categories']=a
```

In [411]:

```
res_table_state.head()
```

Out[411]:

	prob_0	prob_1	categories
AR	0.068966	0.931034	AR
FL	0.193750	0.806250	FL
MO	0.154472	0.845528	MO
NC	0.154930	0.845070	NC
IN	0.140181	0.859819	IN

Training Based on Response Table

Train Data

In [412]:

```
train_coded_state=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [413]:

```
temp_0=[]
temp_1=[]
for cat in X_train_1["school_state"].values:
    if cat in res_table_state["categories"].values:
        temp_0.append(res_table_state.loc[cat,"prob_0"])
        temp_1.append(res_table_state.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [414]:

```
train_coded_state["prob_0"]=temp_0
```

```
train_coded_state["prob_0"]=temp_0
train_coded_state["prob_1"]=temp_1
```

In [415]:

```
train_coded_state.shape
```

Out[415]:

```
(13467, 2)
```

CV Data

In [417]:

```
cv_coded_state=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [418]:

```
temp_0=[]
temp_1=[]
for cat in X_cv_1["school_state"].values:
    if cat in res_table_state["categories"].values:
        temp_0.append(res_table_state.loc[cat,"prob_0"])
        temp_1.append(res_table_state.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [419]:

```
cv_coded_state["prob_0"]=temp_0
cv_coded_state["prob_1"]=temp_1
```

In [420]:

```
cv_coded_state.shape
```

Out[420]:

```
(6633, 2)
```

Test Data

In [421]:

```
test_coded_state=pd.DataFrame(columns=["prob_0","prob_1"])
```

In [422]:

```
temp_0=[]
temp_1=[]
for cat in X_test_1["school_state"].values:
    if cat in res_table_state["categories"].values:
        temp_0.append(res_table_state.loc[cat,"prob_0"])
        temp_1.append(res_table_state.loc[cat,"prob_1"])
    else:
        temp_0.append(0.5)
        temp_1.append(0.5)
```

In [423]:

```
test_coded_state["prob_0"]=temp_0
test_coded_state["prob_1"]=temp_1
```

In [424]:

```
test_coded_state.shape
```

Out[424]:

```
(9900, 2)
```

Vectorizing Text Data

Average word2vector(avg w2v)

In [425]:

```
#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 [426]:

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

```
Loading Glove Model
```

```
2196017it [02:51, 12768.94it/s]
```

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

In [427]:

```
words = []
for i in X_train_1["preprocessed_essays"]:
    words.extend(i.split(' '))
```

In [428]:

```
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 1872146
the unique words in the corpus 24784
The number of words that are present in both glove vectors and our corpus 23066 ( 93.068 %)
word 2 vec length 23066
```

In [429]:

```
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 [430]:

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

Train Essays

In [431]:

```
# 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_1["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(type(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 13467/13467
[00:04<00:00, 3181.39it/s]
```

```
<class 'list'>
13467
300
```

Cross-Validation Essays

In [432]:

```
# 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_1["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%|████████████████████████████████████████████████████████████████████████████| 6633/6633
```

```
[00:02<00:00, 3021.15it/s]
```

```
6633
300
```

Test Essays

In [433]:

```
# 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_1["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%|████████████████████████████████████████████████████████████████████████████████| 9900/9900
[00:03<00:00, 3063.10it/s]
```

```
9900
300
```

Train Titles

In [434]:

```
# 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_1["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%|█████████████████████████████████████████████████████████████████████████████| 13467/13467
[00:00<00:00, 64435.74it/s]
```

```
13467
300
```

Cross-Validation Titles

In [435]:

```
# 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_1["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%|████████████████████████████████████████████████████████████████████████████| 6633/6633
[00:00<00:00, 64404.22it/s]
```

```
6633
300
```

Test Titles

In [436]:

```
# 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_1["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%|████████████████████████████████████████████████████████████████████████████| 9900/9900
[00:00<00:00, 65131.40it/s]
```

```
9900
300
```

Tf-idf weighted W2V(Using Pretrained Model for finding the tf-idf weighted word2vec)

Train Essays

In [437]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train_1["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 [438]:

```
# 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_1["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%|███████████████████████████████████████████████████████████████████████████| 13467/13467 [00:
23<00:00, 569.87it/s]
```

```
13467
300
```

Cross-Validation Essays

In [439]:

```
# 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_1["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%|█████████████████████████████████████████████████████████████████████████████| 6633/6633
[00:11<00:00, 563.19it/s]
```

```
6633
300
```

Test Essays

In [440]:

```
# 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_1["preprocessed_essays"]): # for each sentence
for sentence in tqdm(X_test_1["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%|████████████████████████████████████████████████████████████████████████████████| 9900/9900
[00:18<00:00, 533.63it/s]
```

```
9900
300
```

Train Titles

In [441]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train_1["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 [442]:

```
# 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_1["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%|████████████████████████████████████████████████████████████████████████████| 13467/13467
[00:00<00:00, 35342.10it/s]
```

```
13467
300
```

Cross-Validation Titles

In [443]:

```
# 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_1["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%|███████████████████████████████████████████████████████████████████████████| 6633/6633
[00:00<00:00, 34369.39it/s]
```

```
6633
300
```

Test titles

In [444]:

```
# 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_1["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%|███████████████████████████████████████████████████████████████████████████| 9900/9900
[00:00<00:00, 36391.82it/s]
```

```
9900
300
```

Making numerical features hstack compatible

Train

In [445]:

```
price_train_1=X_train_1['price'].values.reshape(1,-1)
print(price_train_1.shape)
```

```
price_train_1=price_train_1.reshape(-1,1)
print(price_train_1.shape)
```

```
(1, 13467)
(13467, 1)
```

In [446]:

```
quantity_train_1=X_train_1['quantity'].values.reshape(1,-1)
print(quantity_train_1.shape)

quantity_train_1=quantity_train_1.reshape(-1,1)
print(quantity_train_1.shape)
```

```
(1, 13467)
(13467, 1)
```

In [447]:

```
tnp_train_1=X_train_1["teacher_number_of_previously_posted_projects"].values.reshape(1,-1)
print(tnp_train_1.shape)

tnp_train_1=tnp_train_1.reshape(-1,1)
print(tnp_train_1.shape)
```

```
(1, 13467)
(13467, 1)
```

Cross-Validation

In [448]:

```
price_cv_1=X_cv_1['price'].values.reshape(1,-1)
print(price_cv_1.shape)

price_cv_1=price_cv_1.reshape(-1,1)
print(price_cv_1.shape)
```

```
(1, 6633)
(6633, 1)
```

In [449]:

```
quantity_cv_1=X_cv_1['quantity'].values.reshape(1,-1)
print(quantity_cv_1.shape)

quantity_cv_1=quantity_cv_1.reshape(-1,1)
print(quantity_cv_1.shape)
```

```
(1, 6633)
(6633, 1)
```

In [450]:

```
tnp_cv_1=X_cv_1["teacher_number_of_previously_posted_projects"].values.reshape(1,-1)
print(tnp_cv_1.shape)

tnp_cv_1=tnp_cv_1.reshape(-1,1)
print(tnp_cv_1.shape)
```

```
(1, 6633)
(6633, 1)
```

Test

Test

In [451]:

```
price_test_1=X_test_1['price'].values.reshape(1,-1)
print(price_test_1.shape)

price_test_1=price_test_1.reshape(-1,1)
print(price_test_1.shape)
```

```
(1, 9900)
(9900, 1)
```

In [452]:

```
quantity_test_1=X_test_1['quantity'].values.reshape(1,-1)
print(quantity_test_1.shape)

quantity_test_1=quantity_test_1.reshape(-1,1)
print(quantity_test_1.shape)
```

```
(1, 9900)
(9900, 1)
```

In [453]:

```
tnp_test_1=X_test_1["teacher_number_of_previously_posted_projects"].values.reshape(1,-1)
print(tnp_test_1.shape)

tnp_test_1=tnp_test_1.reshape(-1,1)
print(tnp_test_1.shape)
```

```
(1, 9900)
(9900, 1)
```

Applying Random Forest

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

In [455]:

```
from scipy.sparse import hstack
X_tr_3=hstack((train_coded_cat ,train_coded_subcat ,train_coded_prefix, train_coded_grade
,train_coded_state,avg_w2v_vectors_train, avg_w2v_vectors_title_train,price_train_1,
quantity_train_1 ,tnp_train_1)).tocsr()

X_cv_3=hstack((cv_coded_cat ,cv_coded_subcat ,cv_coded_prefix, cv_coded_grade ,cv_coded_state,avg_w
2v_vectors_cv, avg_w2v_vectors_title_cv,price_cv_1 ,quantity_cv_1, tnp_cv_1)).tocsr()

X_te_3=hstack((test_coded_cat ,test_coded_subcat ,test_coded_prefix, test_coded_grade
,test_coded_state,avg_w2v_vectors_test, avg_w2v_vectors_title_test,price_test_1, quantity_test_1 ,
tnp_test_1)).tocsr()
```

In [458]:

```
#checking the final matrix are of same dimension or not
print(X_tr_3.shape,y_train_1.shape)
print("="*50)
print(X_cv_3.shape,y_cv_1.shape)
print("="*50)
print(X_te_3.shape,y_test_1.shape)
```

```
(13467, 613) (13467,)
==================================================
(6633, 613) (6633,)
==================================================
(9900, 613) (9900,)
```

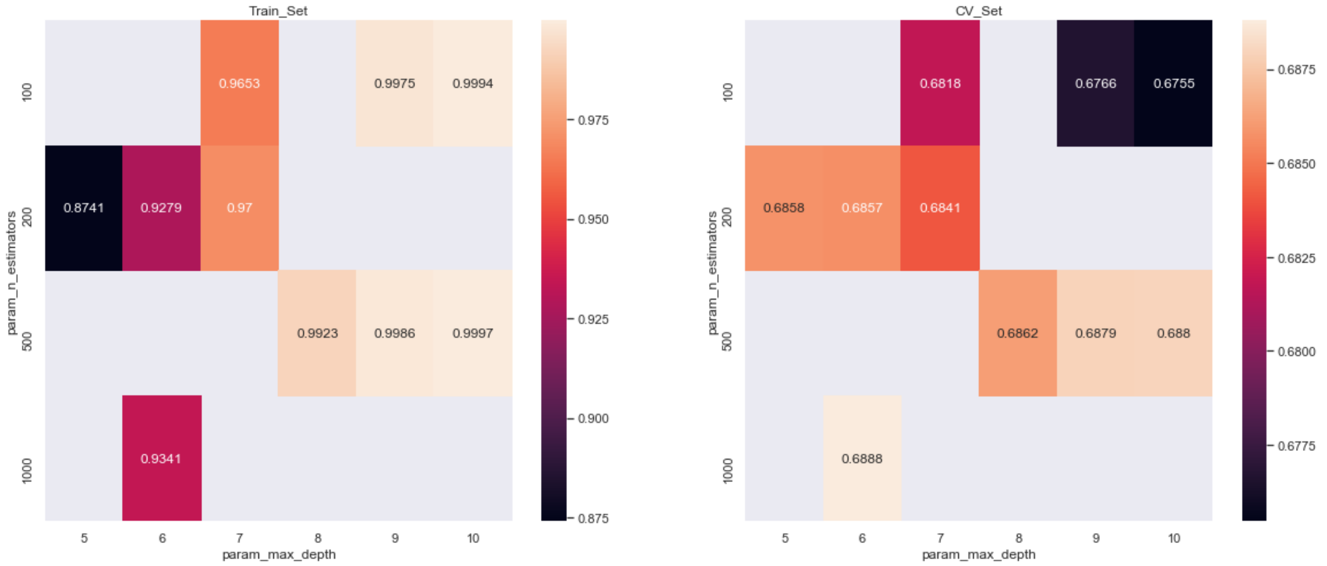
finding best Hyperparameters using RandomizedSearchCV

In [459]:

```
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(class_weight='balanced')
parameters={'n_estimators':[100, 150, 200, 300, 500, 1000], 'max_depth':[5, 6, 7, 8, 9, 10]}

clf=RandomizedSearchCV(rf,parameters, cv=3, scoring='roc_auc', return_train_score=True)

set3=clf.fit(X_tr_3,y_train_1)
```

In [460]:

```
import seaborn as sns
sns.set()
df3=pd.DataFrame(clf.cv_results_).groupby(['param_n_estimators', 'param_max_depth']).max().unstack
()[['mean_test_score', 'mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,8))
sns.heatmap(df3.mean_train_score,annot=True, fmt='.4g', ax=ax[0])
sns.heatmap(df3.mean_test_score,annot=True, fmt='.4g', ax=ax[1])
ax[0].set_title("Train_Set")
ax[1].set_title("CV_Set")
plt.show()
```



In [461]:

```
print(clf.best_estimator_)

print(clf.score(X_tr_3,y_train_1))
print(clf.score(X_cv_3,y_cv_1))
```

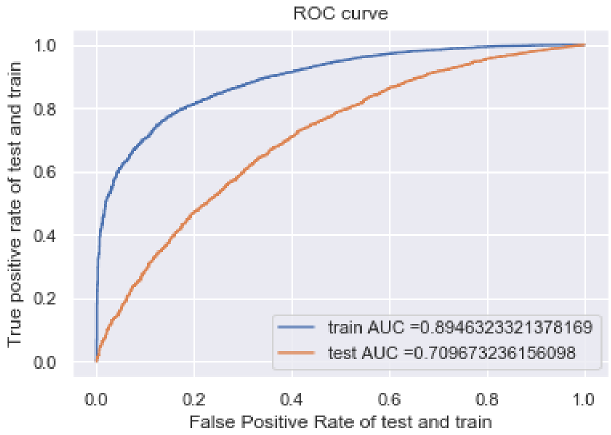
```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=6, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=1000, n_jobs=None, oob_score=False,
            random_state=None, verbose=0, warm_start=False)
0.8931049146055727
0.7068176910274842
```

Testing on Test Data(using our max_depth=6 and n_estimators=1000)

Testing on Test Data(using our max_depth=6 and n_estimators=1000)

In [463]:

```
rf = RandomForestClassifier(n_estimators=1000, max_depth=6,class_weight='balanced')

rf.fit(X_tr_3, y_train_1)
train_predict=rf.predict_proba(X_tr_3)[:,1]
test_predict= rf.predict_proba(X_te_3)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train_1,train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test_1,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(True)
plt.show()
```



Confusion Matrix

In [464]:

```
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   1.791815
1  0.0  0.000088   0.791815
2  0.0  0.068890   0.676125
(2232, 3)
```

In [465]:

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

In [467]:

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

In [468]:

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

df.head(3)
```

Out[468]:

	fpr	tpr	threshold	Specificty	Value
560	0.146236	0.773497	0.522849	0.853764	0.660384
558	0.145753	0.773058	0.523014	0.854247	0.660383
562	0.146718	0.773760	0.522826	0.853282	0.660236

In [469]:

```
index = df.Value.argmax()
```

In [471]:

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

```
0.5228488556925284
```

In [472]:

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

```
[0.]
```

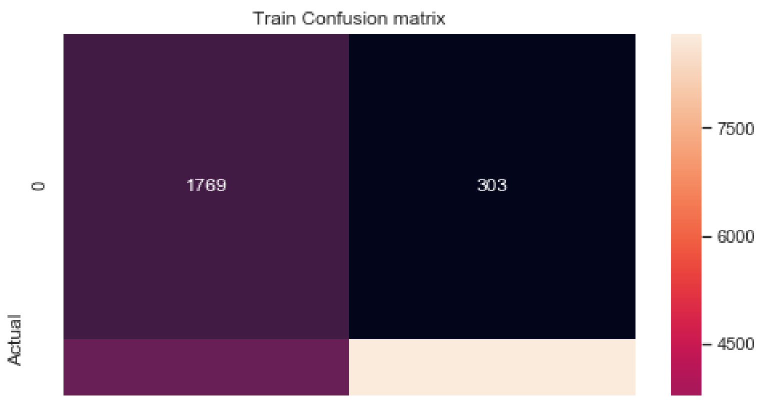
In [473]:

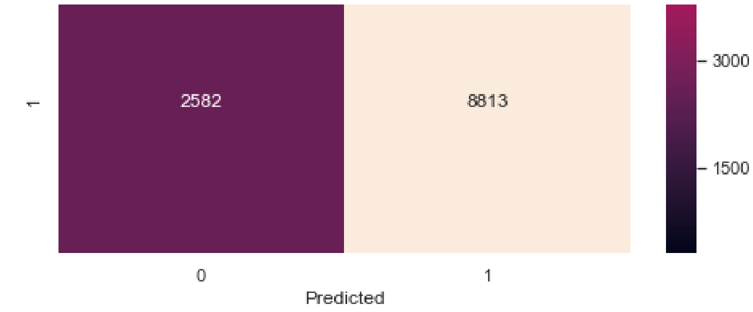
```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("confusion matrix")
cm=confusion_matrix(y_train_1, y_predict_thres)
print(cm)

#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")
```

```
Threshold 0.5228488556925284
confusion matrix
[[1769  303]
 [2582 8813]]
```

Out[473]:

```
Text(0.5, 39.5, 'Predicted')
```





Test Data

In [474]:

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

```
[1.]
```

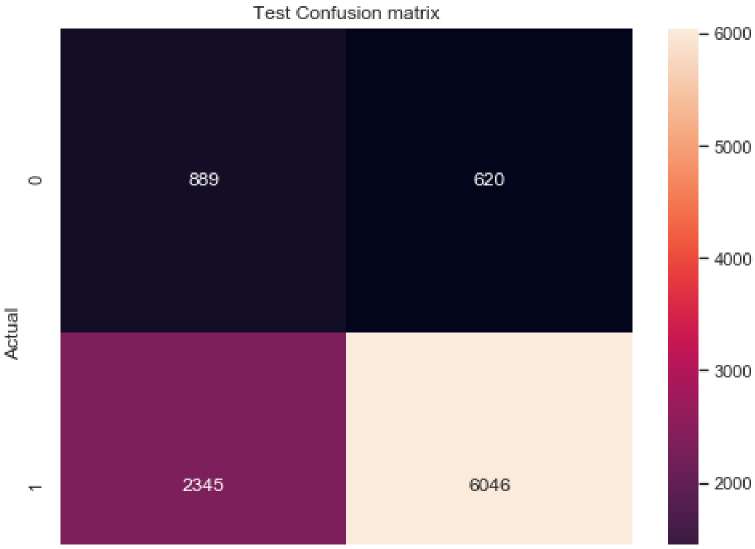
In [475]:

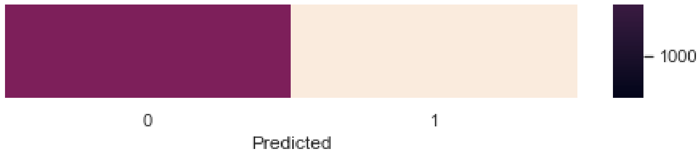
```
from sklearn.metrics import confusion_matrix
print("Threshold",a)

print("Test confusion matrix")
cm1=confusion_matrix(y_test_1, y_predict_thres)
print(cm1)

#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")
```

```
Threshold 0.5228488556925284
Test confusion matrix
[[ 889  620]
 [2345 6046]]
```

Out[475]:

```
Text(0.5, 39.5, 'Predicted')
```





Set 4: Categorical Features,Numerical Features+Preprocessed Essay(tf-idf W2Vec)+Preprocessed Title(tf-idf W2Vec)

In [476]:

```
from scipy.sparse import hstack
X_tr_4=hstack((train_coded_cat ,train_coded_subcat ,train_coded_prefix, train_coded_grade
,train_coded_state,tfidf_w2v_vectors_train, tfidf_w2v_vectors_title_train,price_train_1,
quantity_train_1 ,tnp_train_1)).tocsr()

X_cv_4=hstack((cv_coded_cat ,cv_coded_subcat ,cv_coded_prefix, cv_coded_grade ,cv_coded_state,tfidf
_w2v_vectors_cv, tfidf_w2v_vectors_title_cv,price_cv_1 ,quantity_cv_1, tnp_cv_1)).tocsr()

X_te_4=hstack((test_coded_cat ,test_coded_subcat ,test_coded_prefix, test_coded_grade
,test_coded_state,tfidf_w2v_vectors_test, tfidf_w2v_vectors_title_test,price_test_1,
quantity_test_1 ,tnp_test_1)).tocsr()
```

In [477]:

```
#checking the final matrix are of same dimension or not
print(X_tr_4.shape,y_train_1.shape)
print("="*50)
print(X_cv_4.shape,y_cv_1.shape)
print("="*50)
print(X_te_4.shape,y_test_1.shape)
```

```
(13467, 613) (13467,)
==================================================
(6633, 613) (6633,)
==================================================
(9900, 613) (9900,)
```

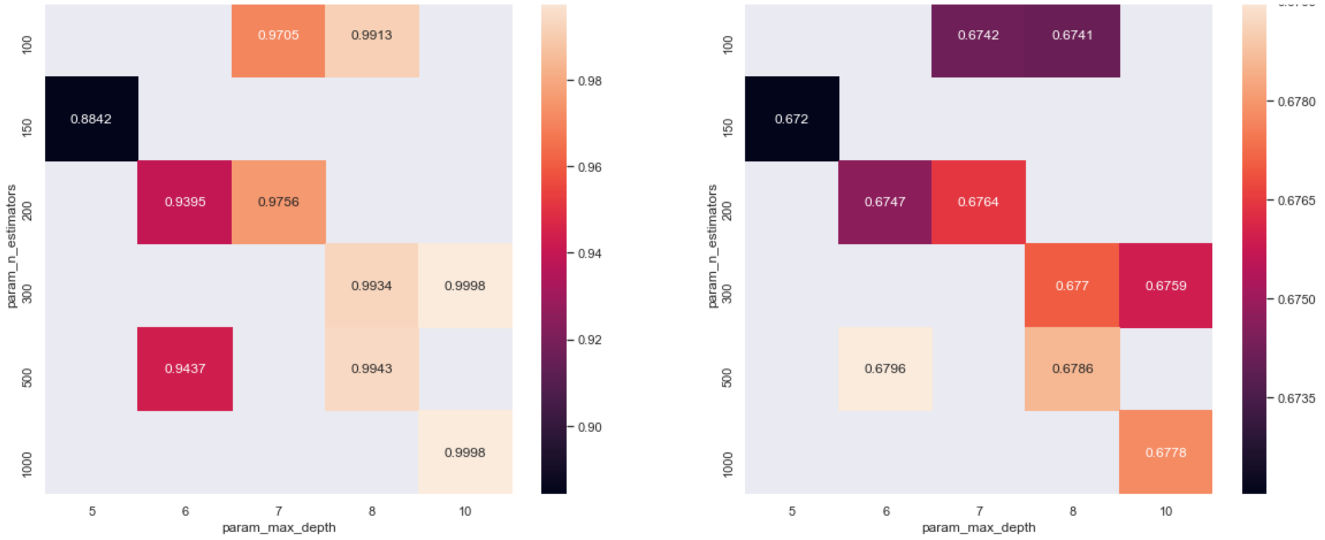
finding best Hyperparameters using RandomizedSearchCV

In [479]:

```
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(class_weight='balanced')
parameters={'n_estimators':[100, 150, 200, 300, 500, 1000], 'max_depth':[5, 6, 7, 8, 9, 10]}

clf=RandomizedSearchCV(rf,parameters, cv=2, scoring='roc_auc', return_train_score=True)

set4=clf.fit(X_tr_4,y_train_1)
```

In [480]:

```
import seaborn as sns
sns.set()
df4=pd.DataFrame(clf.cv_results_).groupby(['param_n_estimators', 'param_max_depth']).max().unstack
()[['mean_test_score', 'mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,8))
sns.heatmap(df4.mean_train_score,annot=True, fmt='.4g', ax=ax[0])
sns.heatmap(df4.mean_test_score,annot=True, fmt='.4g', ax=ax[1])
ax[0].set_title("Train_Set")
ax[1].set_title("CV_Set")
plt.show()
```



In [481]:

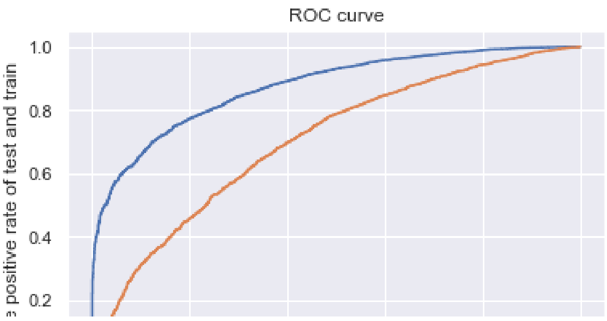
```
print(clf.best_estimator_)

print(clf.score(X_tr_4,y_train_1))
print(clf.score(X_cv_4,y_cv_1))
```

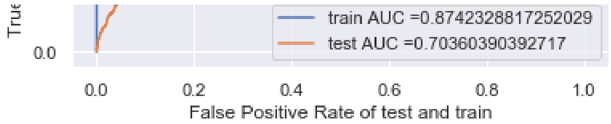
```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=6, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=500, n_jobs=None, oob_score=False,
            random_state=None, verbose=0, warm_start=False)
0.8731362058479214
0.6952506176418146
```

Testing on Test Data(using our max_depth=6 and n_estimators=500)

In [482]:

```
rf = RandomForestClassifier(n_estimators=500, max_depth=6,class_weight='balanced')

rf.fit(X_tr_4, y_train_1)
train_predict=rf.predict_proba(X_tr_4)[:,1]
test_predict= rf.predict_proba(X_te_4)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train_1,train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test_1,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(True)
plt.show()
```





Confusion Matrix

In [483]:

```
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   1.831405
1  0.0  0.000088   0.831405
2  0.0  0.222905   0.641687
(2510, 3)
```

In [484]:

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

In [485]:

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

In [486]:

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

df.head(3)
```

Out[486]:

	fpr	tpr	threshold	Specificty	Value
634	0.167954	0.749013	0.515942	0.832046	0.623213
630	0.166988	0.747696	0.516339	0.833012	0.622840
638	0.169402	0.749803	0.515758	0.830598	0.622785

In [487]:

```
index = df.Value.argmax()
```

In [488]:

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

```
0.5159416399754889
```

In [489]:

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

```
[1.]
```

In [490]:

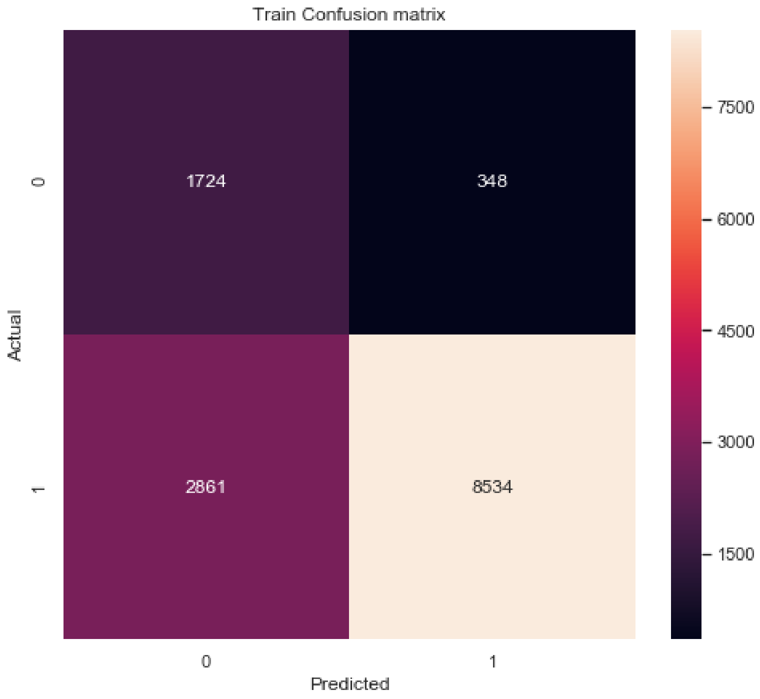
In [490]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("confusion matrix")
cm=confusion_matrix(y_train_1, y_predict_thres)
print(cm)

#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")
```

```
Threshold 0.5159416399754889
confusion matrix
[[1724  348]
 [2861 8534]]
```

Out[490]:

```
Text(0.5, 39.5, 'Predicted')
```



Test Data

In [491]:

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

```
[1.]
```

In [492]:

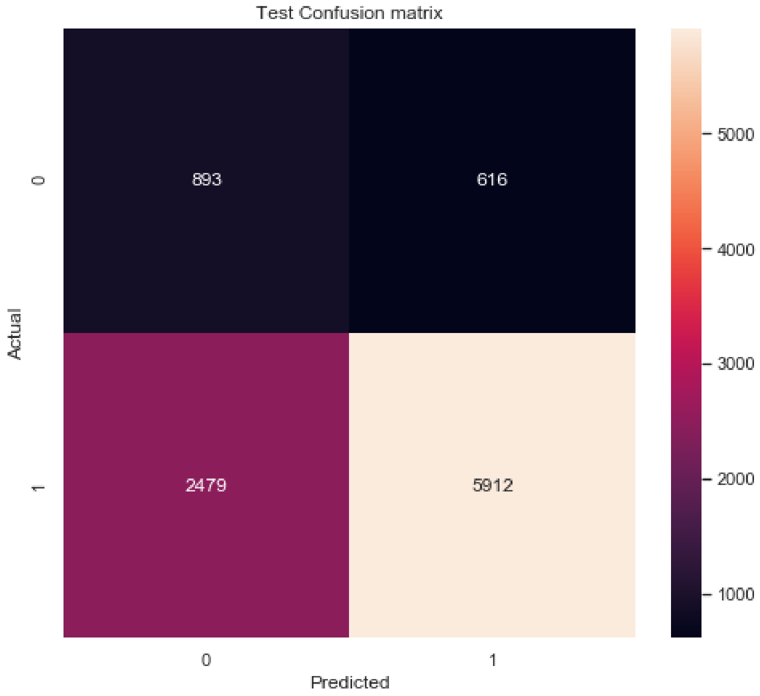
```
from sklearn.metrics import confusion_matrix
print("Threshold",a)

print("Test confusion matrix")
cm1=confusion_matrix(y_test_1, y_predict_thres)
print(cm1)
```

```
#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")
```

```
Threshold 0.5159416399754889
Test confusion matrix
[[ 893  616]
 [2479 5912]]
```

Out[492]:

```
Text(0.5, 39.5, 'Predicted')
```



Applying XGBoost

XGBoost stands for eXtreme Gradient Boosting, it is an implementation of gradient boosting machines

Bag of words(BoW)

Preprocessed Essay

In [493]:

```
model_essay_bow =  CountVectorizer(min_df=10)
model_essay_bow.fit(X_train_1["preprocessed_essays"])


train_bow_essay1 = model_essay_bow.transform(X_train_1["preprocessed_essays"])
print("Shape of matrix ",train_bow_essay1.shape)
print("="*50)
cv_bow_essay1=model_essay_bow.transform(X_cv_1["preprocessed_essays"]) #BoW of CV
print("Shape of matrix ",cv_bow_essay1.shape)
print("="*50)
test_bow_essay1 = model_essay_bow.transform(X_test_1["preprocessed_essays"]) #BoW of Test
print("Shape of matrix ",test_bow_essay1.shape)
```

```
Shape of matrix  (13467, 6956)
==================================================
Shape of matrix  (6633, 6956)
==================================================
Shape of matrix  (9900, 6956)
```

Preprocessed Title

In [494]:

```
model_title_bow =  CountVectorizer(min_df=10)
model_title_bow.fit(X_train_1["preprocessed_title"])


train_bow_title1 = model_title_bow.transform(X_train_1["preprocessed_title"])
print("Shape of matrix ",train_bow_title1.shape)
print("="*50)
cv_bow_title1=model_title_bow.transform(X_cv_1["preprocessed_title"]) #BoW of test
print("Shape of matrix ",cv_bow_title1.shape)
print("="*50)
test_bow_title1 = model_title_bow.transform(X_test_1["preprocessed_title"]) #BoW of Cross
Validation
print("Shape of matrix ",test_bow_title1.shape)
```

```
Shape of matrix  (13467, 751)
==================================================
Shape of matrix  (6633, 751)
==================================================
Shape of matrix  (9900, 751)
```

Set 1: Categorical Features,Numerical Features+Preprocessed Essay(BOW)+Preprocessed Title(BOW)

In [495]:

```
from scipy.sparse import hstack
X_tr_11=hstack((train_coded_cat ,train_coded_subcat ,train_coded_prefix, train_coded_grade
,train_coded_state,train_bow_essay1, train_bow_title1,price_train_1, quantity_train_1 ,tnp_train_1)
).tocsr()

X_cv_11=hstack((cv_coded_cat ,cv_coded_subcat ,cv_coded_prefix, cv_coded_grade
,cv_coded_state,cv_bow_essay1, cv_bow_title1,price_cv_1 ,quantity_cv_1, tnp_cv_1)).tocsr()

X_te_11=hstack((test_coded_cat ,test_coded_subcat ,test_coded_prefix, test_coded_grade
,test_coded_state,test_bow_essay1, test_bow_title1,price_test_1, quantity_test_1
,tnp_test_1)).tocsr()
```

In [496]:

```
#checking the final matrix are of same dimension or not
print(X_tr_11.shape,y_train_1.shape)
print("="*50)
print(X_cv_11.shape,y_cv_1.shape)
print("="*50)
print(X_te_11.shape,y_test_1.shape)
```

```
(13467, 7720) (13467,)
==================================================
(6633, 7720) (6633,)
==================================================
(9900, 7720) (9900,)
```

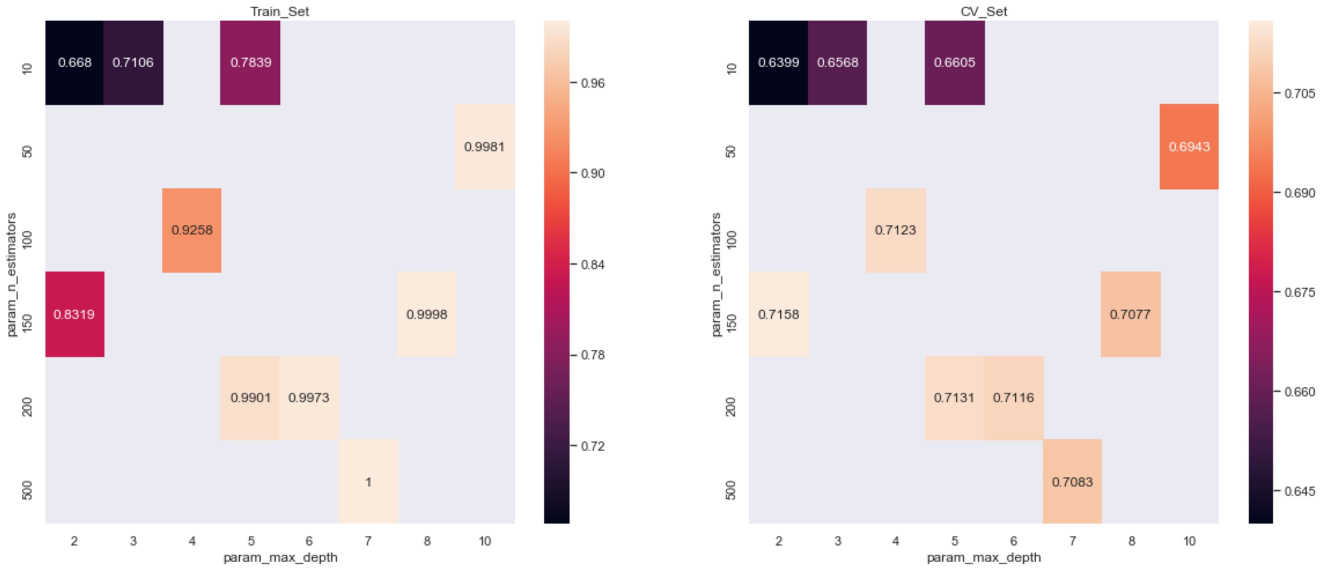
finding best Hyperparameters Using RandomizedSearchCV

In [497]:

```
import xgboost as xgb
import xgboost as xgb
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
clf_xgb = xgb.XGBClassifier()
parameters={'n_estimators':[10, 50, 100, 150, 200, 300, 500], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 1
0]}

clf=RandomizedSearchCV(clf_xgb,parameters, cv=2, scoring='roc_auc', return_train_score=True)

set1=clf.fit(X_tr_11,y_train_1)
```

In [498]:

```
import seaborn as sns
sns.set()
df11=pd.DataFrame(clf.cv_results_).groupby(['param_n_estimators',
'param_max_depth']).max().unstack()[['mean_test_score', 'mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,8))
sns.heatmap(df11.mean_train_score,annot=True, fmt='.4g', ax=ax[0])
sns.heatmap(df11.mean_test_score,annot=True, fmt='.4g', ax=ax[1])
ax[0].set_title("Train_Set")
ax[1].set_title("CV_Set")
plt.show()
```



In [499]:

```
print(clf.best_estimator_)

print(clf.score(X_tr_11,y_train_1))
print(clf.score(X_cv_11,y_cv_1))
```

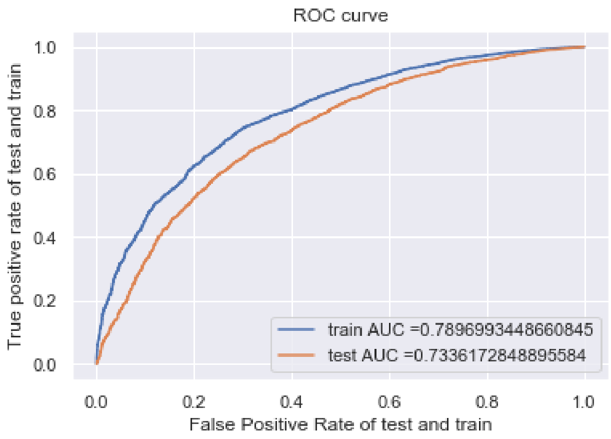
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=2, min_child_weight=1, missing=None,
       n_estimators=150, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
0.7896993448660845
0.7298988775157845
```

Testing on Test Data(using our max_depth=2 and n_estimators=150)

In [501]:

```
clf_xgb = xgb.XGBClassifier(n_estimators=150, max_depth=2,class_weight='balanced',learning_rate=0.1
)

clf_xgb.fit(X_tr_11, y_train_1)
```

```
train_predict=clf_xgb.predict_proba(X_tr_11)[:,1]
test_predict= clf_xgb.predict_proba(X_te_11)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train_1,train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test_1,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(True)
plt.show()
```



Confusion Matrix

In [502]:

```
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   1.985904
1  0.0  0.000088   0.985904
2  0.0  0.005178   0.967838
(3067, 3)
```

In [503]:

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

In [504]:

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

In [505]:

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

df.head(3)
```

Out[505]:

	fpr	tpr	threshold	Specificty	Value
1204	0.290058	0.733304	0.828133	0.709942	0.520603
1213	0.292954	0.736288	0.827575	0.707046	0.520590

	fpr	tpr	threshold	Specificty	Value
1202	0.289575	0.732690	0.828266	0.710425	0.520521

In [506]:

```
index = df.Value.argmax()
```

In [507]:

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

```
0.82813334
```

In [508]:

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

```
[1.]
```

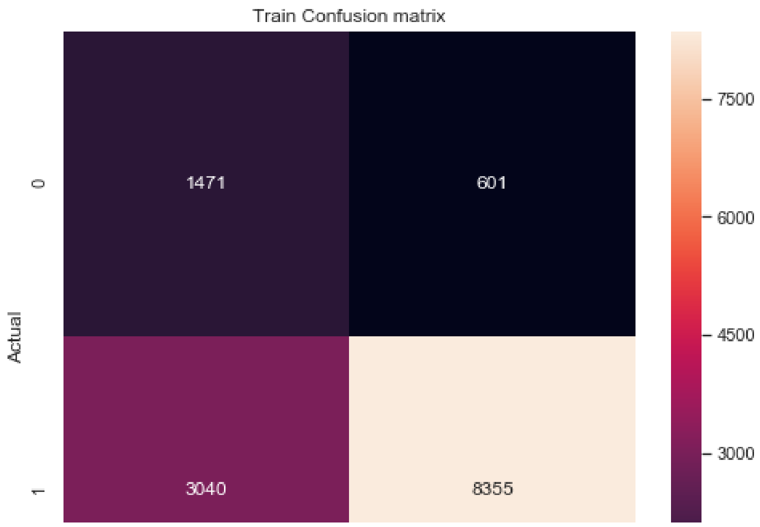
In [509]:

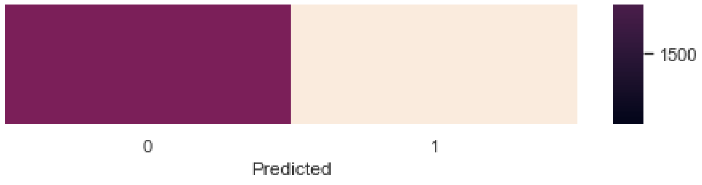
```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("confusion matrix")
cm=confusion_matrix(y_train_1, y_predict_thres)
print(cm)



#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")
```

```
Threshold 0.82813334
confusion matrix
[[1471  601]
 [3040 8355]]
```

Out[509]:

```
Text(0.5, 39.5, 'Predicted')
```





Test Data

In [510]:

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

```
[1.]
```

In [511]:

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

```
Threshold 0.82813334
```

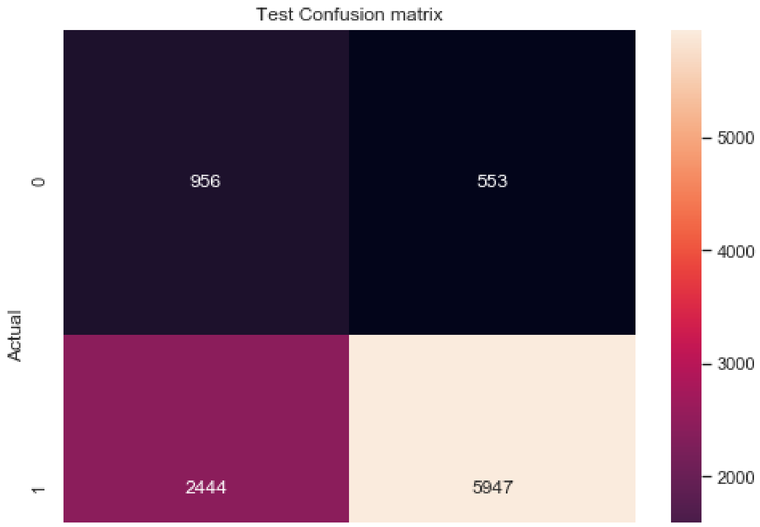
In [512]:

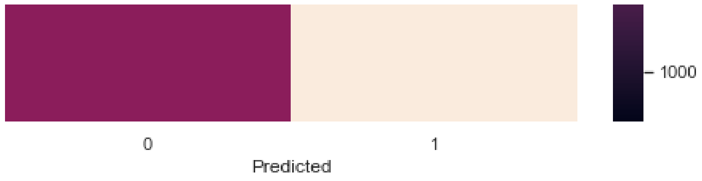
```
print("Test confusion matrix")
cm1=confusion_matrix(y_test_1, y_predict_thres)
print(cm1)


#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")
```

```
Test confusion matrix
[[ 956  553]
 [2444 5947]]
```

Out[512]:

```
Text(0.5, 39.5, 'Predicted')
```





Set 2: Categorical Features,Numerical Features+Preprocessed Essay(tf-idf)+Preprocessed Title(tf-idf)

Tf-idf vectorizer

Tf-idf of Project_Essays

In [514]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
model_essay_tfidf = TfidfVectorizer(min_df=10)
model_essay_tfidf.fit(X_train_1["preprocessed_essays"])

train_tfidf_essay1=model_essay_tfidf.transform(X_train_1["preprocessed_essays"])
print("Shape of matrix ",train_tfidf_essay1.shape)
print("="*50)
cv_tfidf_essay1=model_essay_tfidf.transform(X_cv_1["preprocessed_essays"]) #tfidf of CV
print("Shape of matrix ",cv_tfidf_essay1.shape)
print("="*50)
test_tfidf_essay1 = model_essay_tfidf.transform(X_test_1["preprocessed_essays"]) #tfidf of Test
print("Shape of matrix ",test_tfidf_essay1.shape)
```

```
Shape of matrix  (13467, 6956)
==================================================
Shape of matrix  (6633, 6956)
==================================================
Shape of matrix  (9900, 6956)
```

Tf-idf of Project_Title

In [515]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
model_title_tfidf = TfidfVectorizer(min_df=10)
model_title_tfidf.fit(X_train_1["preprocessed_title"])

train_tfidf_title1=model_title_tfidf.transform(X_train_1["preprocessed_title"])
print("Shape of matrix ",train_tfidf_title1.shape)
print("="*50)
cv_tfidf_title1=model_title_tfidf.transform(X_cv_1["preprocessed_title"]) #tfidf of CV
print("Shape of matrix ",cv_tfidf_title1.shape)
print("="*50)
test_tfidf_title1 = model_title_tfidf.transform(X_test_1["preprocessed_title"]) #tfidf of Test
print("Shape of matrix ",test_tfidf_title1.shape)
```

```
Shape of matrix  (13467, 751)
==================================================
Shape of matrix  (6633, 751)
==================================================
Shape of matrix  (9900, 751)
```

In [519]:

```
from scipy.sparse import hstack
X_tr_22=hstack((train_coded_cat ,train_coded_subcat ,train_coded_prefix, train_coded_grade
,train_coded_state,train_tfidf_essay1, train_tfidf_title1,price_train_1, quantity_train_1
,tnp_train_1)).tocsr()

X_cv_22=hstack((cv_coded_cat ,cv_coded_subcat ,cv_coded_prefix, cv_coded_grade
,cv_coded_state,cv_tfidf_essay1, cv_tfidf_title1,price_cv_1, quantity_cv_1 ,tnp_cv_1)).tocsr()
```

```
,cv_coded_state,cv_tfidf_essay1, cv_tfidf_title1,price_cv_1 ,quantity_cv_1, tnp_cv_1)).tocsr()

X_te_22=hstack((test_coded_cat ,test_coded_subcat ,test_coded_prefix, test_coded_grade
,test_coded_state,test_tfidf_essay1 ,test_tfidf_title1,price_test_1, quantity_test_1 ,tnp_test_1))
.tocsr()
```

In [520]:

```
#checking the final matrix are of same dimension or not
print(X_tr_22.shape,y_train_1.shape)
print("="*50)
print(X_cv_22.shape,y_cv_1.shape)
print("="*50)
print(X_te_22.shape,y_test_1.shape)
```

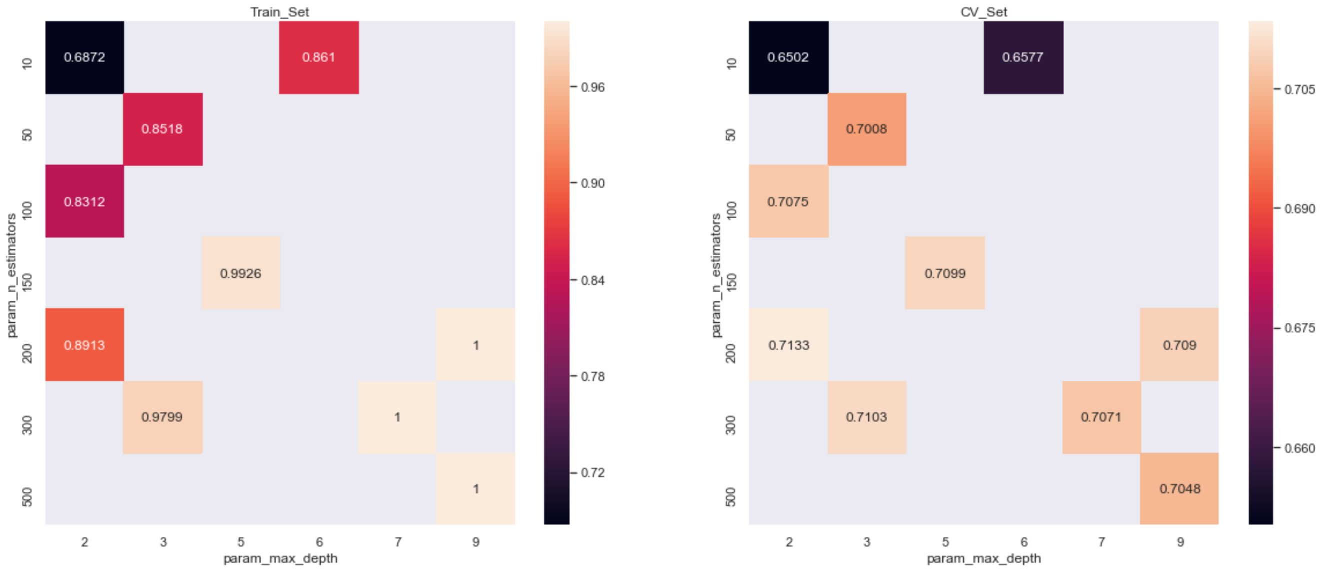
```
(13467, 7720) (13467,)
==================================================
(6633, 7720) (6633,)
==================================================
(9900, 7720) (9900,)
```

In [521]:

```
import xgboost as xgb
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
clf_xgb = xgb.XGBClassifier()
parameters={'n_estimators':[10, 50, 100, 150, 200, 300, 500], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 1
0]}

clf=RandomizedSearchCV(clf_xgb,parameters, cv=2, scoring='roc_auc', return_train_score=True)

set22=clf.fit(X_tr_22,y_train_1)
```

In [522]:

```
import seaborn as sns
sns.set()
df22=pd.DataFrame(clf.cv_results_).groupby(['param_n_estimators',
'param_max_depth']).max().unstack()[['mean_test_score', 'mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,8))
sns.heatmap(df22.mean_train_score,annot=True, fmt='.4g', ax=ax[0])
sns.heatmap(df22.mean_test_score,annot=True, fmt='.4g', ax=ax[1])
ax[0].set_title("Train_Set")
ax[1].set_title("CV_Set")
plt.show()
```



In [523]:

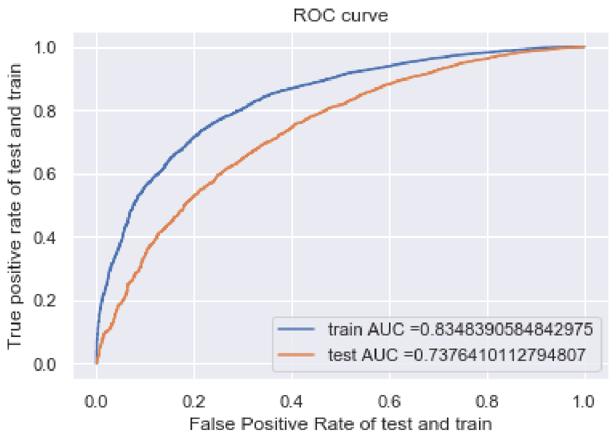
```
print(clf.best_estimator_)

print(clf.score(X_tr_22,y_train_1))
print(clf.score(X_cv_22,y_cv_1))
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=2, min_child_weight=1, missing=None,
       n_estimators=200, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
0.8348390584842975
0.7286923413099448
```

Testing on Test Data(using our max_depth=2 and n_estimators=200)

In [524]:

```
clf_xgb = xgb.XGBClassifier(n_estimators=200, max_depth=2,class_weight='balanced',learning_rate=0.1
)

clf_xgb.fit(X_tr_22, y_train_1)
train_predict=clf_xgb.predict_proba(X_tr_22)[:,1]
test_predict= clf_xgb.predict_proba(X_te_22)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train_1,train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test_1,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(True)
plt.show()
```



Confusion Matrix

In [525]:

```
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   1.992197
1  0.0  0.000088   0.992197
2  0.0  0.011935   0.968164
(2763, 3)
```

In [526]:

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

In [527]:

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

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

df.head(3)
```

Out[527]:

	fpr	tpr	threshold	Specificty	Value
981	0.238417	0.759193	0.825149	0.761583	0.578188
983	0.238900	0.759544	0.825031	0.761100	0.578089
985	0.239382	0.759982	0.824860	0.760618	0.578056

In [528]:

```
index = df.Value.argmax()
```

In [529]:

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

```
0.82514894
```

In [530]:

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

```
[1.]
```

In [531]:

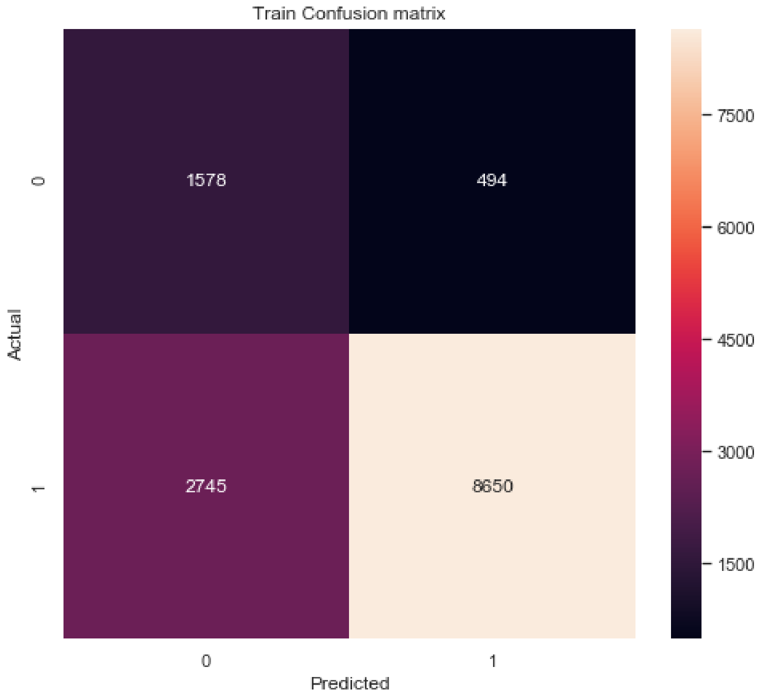
```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("confusion matrix")
cm=confusion_matrix(y_train_1, y_predict_thres)
print(cm)



#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")
```

```
Threshold 0.82514894
confusion matrix
[[1578  494]
 [2745 8650]]
```

Out[531]:

Out[531]:

```
Text(0.5, 39.5, 'Predicted')
```



Test Data

In [532]:

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

```
[1.]
```

In [533]:

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

```
Threshold 0.82514894
```

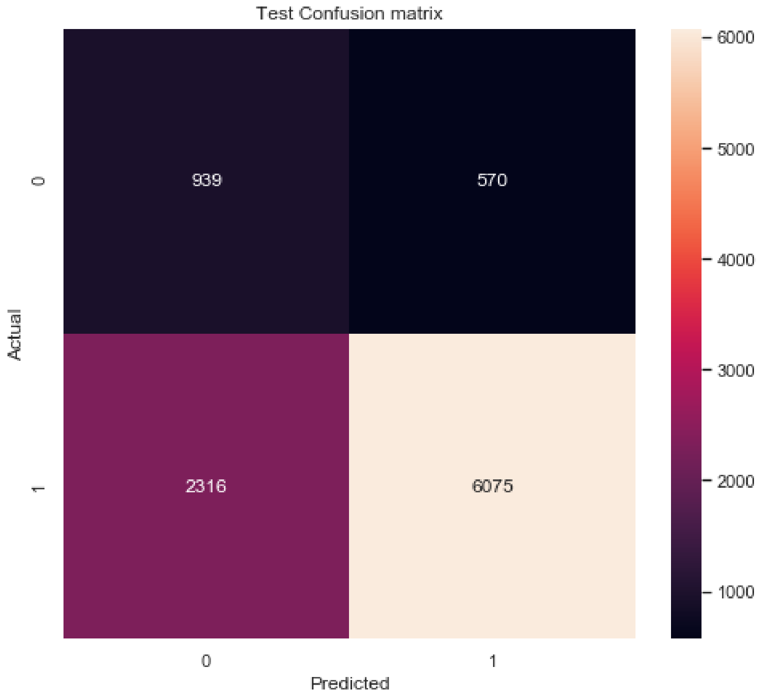
In [534]:

```
print("Test confusion matrix")
cm1=confusion_matrix(y_test_1, y_predict_thres)
print(cm1)


#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")
```

```
Test confusion matrix
[[ 939  570]
 [2316 6075]]
```

Out[534]:

```
Text(0.5, 39.5, 'Predicted')
```



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

In [535]:

```
from scipy.sparse import hstack
X_tr_33=hstack((train_coded_cat ,train_coded_subcat ,train_coded_prefix, train_coded_grade
,train_coded_state,avg_w2v_vectors_train, avg_w2v_vectors_title_train,price_train_1,
quantity_train_1 ,tnp_train_1)).tocsr()

X_cv_33=hstack((cv_coded_cat ,cv_coded_subcat ,cv_coded_prefix, cv_coded_grade
,cv_coded_state,avg_w2v_vectors_cv, avg_w2v_vectors_title_cv,price_cv_1 ,quantity_cv_1, tnp_cv_1)).
tocsr()

X_te_33=hstack((test_coded_cat ,test_coded_subcat ,test_coded_prefix, test_coded_grade
,test_coded_state,avg_w2v_vectors_test, avg_w2v_vectors_title_test,price_test_1, quantity_test_1 ,
tnp_test_1)).tocsr()
```

In [536]:

```
#checking the final matrix are of same dimension or not
print(X_tr_33.shape,y_train_1.shape)
print("="*50)
print(X_cv_33.shape,y_cv_1.shape)
print("="*50)
print(X_te_33.shape,y_test_1.shape)
```

```
(13467, 613) (13467,)
==================================================
(6633, 613) (6633,)
==================================================
(9900, 613) (9900,)
```

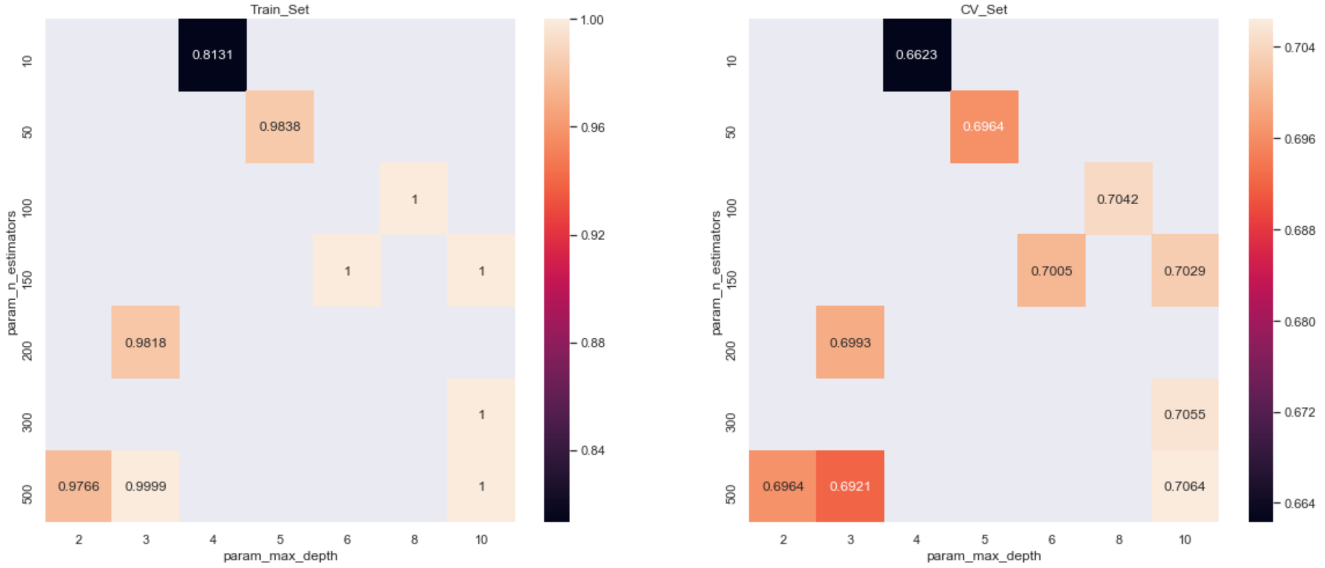
finding best Hyperparameters using RandomizedSearchCV

In [555]:

```
import xgboost as xgb
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
clf_xgb = xgb.XGBClassifier()
parameters={'n_estimators':[10, 50, 100, 150, 200, 300, 500], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 1
0]}

clf=RandomizedSearchCV(clf_xgb,parameters, cv=2, scoring='roc_auc', return_train_score=True)

set33=clf.fit(X_tr_33,y_train_1)
```

In [556]:

```
import seaborn as sns
sns.set()
df33=pd.DataFrame(clf.cv_results_).groupby(['param_n_estimators',
'param_max_depth']).max().unstack()[['mean_test_score', 'mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,8))
sns.heatmap(df33.mean_train_score,annot=True, fmt='.4g', ax=ax[0])
sns.heatmap(df33.mean_test_score,annot=True, fmt='.4g', ax=ax[1])
ax[0].set_title("Train_Set")
ax[1].set_title("CV_Set")
plt.show()
```



In [557]:

```
print(clf.best_estimator_)

print(clf.score(X_tr_33,y_train_1))
print(clf.score(X_cv_33,y_cv_1))
```

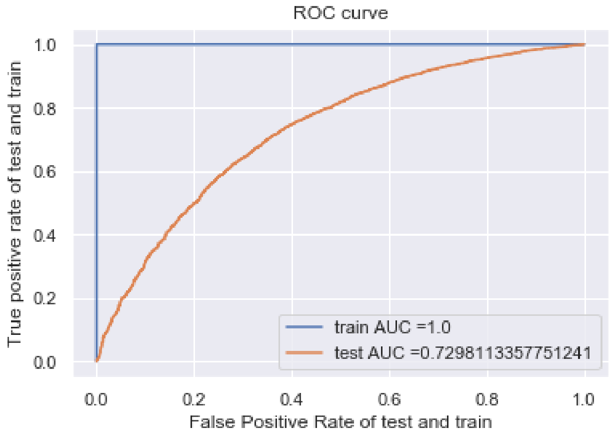
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=10, min_child_weight=1, missing=None,
       n_estimators=500, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
1.0
0.722056392177826
```

Testing on Test Data(using our max_depth=10 and n_estimators=500)

In [558]:

```
clf_xgb = xgb.XGBClassifier(n_estimators=500,
max_depth=10,class_weight='balanced',learning_rate=0.1)

clf_xgb.fit(X_tr_33, y_train_1)
train_predict=clf_xgb.predict_proba(X_tr_33)[:,1]
test_predict= clf_xgb.predict_proba(X_te_33)[:,1]
```

```
train_fpr,train_tpr,train_thresholds= roc_curve(y_train_1,train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test_1,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(True)
plt.show()
```



Confusion Matrix

In [559]:

```
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   1.999994
1  0.0  0.000088   0.999994
2  0.0  0.002194   0.999959
(3068, 3)
```

In [560]:

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

In [561]:

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

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

df.head(3)
```

Out[561]:

	fpr	tpr	threshold	Specificty	Value
3064	0.0	1.000000	0.991963	1.0	1.000000
3063	0.0	0.992190	0.994844	1.0	0.992190
3062	0.0	0.992014	0.994851	1.0	0.992014

In [562]:

```
index = df.Value.argmax()
```

In [563]:

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

```
0.99196285
```

In [564]:

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

```
[1.]
```

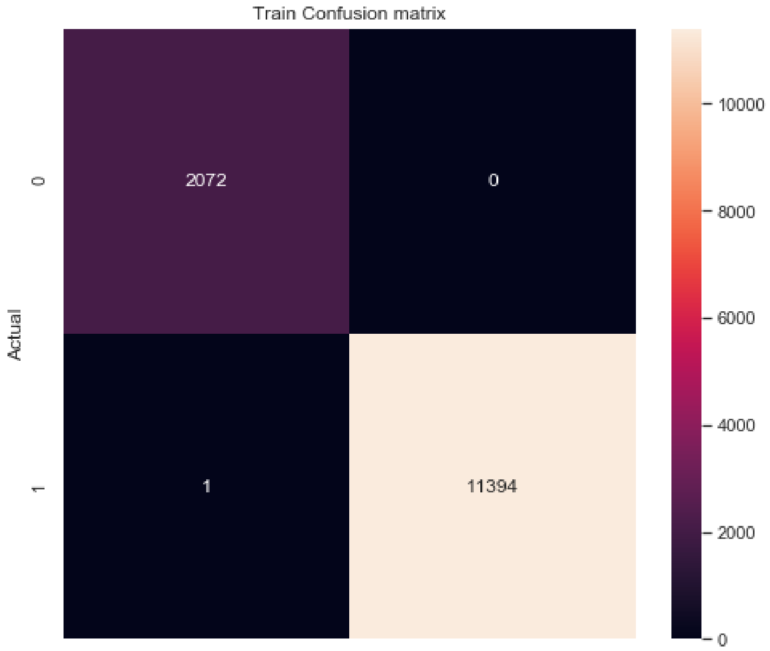
In [565]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("confusion matrix")
cm=confusion_matrix(y_train_1, y_predict_thres)
print(cm)



#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")
```

```
Threshold 0.99196285
confusion matrix
[[ 2072     0]
 [    1 11394]]
```

Out[565]:

```
Text(0.5, 39.5, 'Predicted')
```



Test Data

In [566]:

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

```
[1.]
```

In [567]:

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

```
Threshold 0.99196285
```

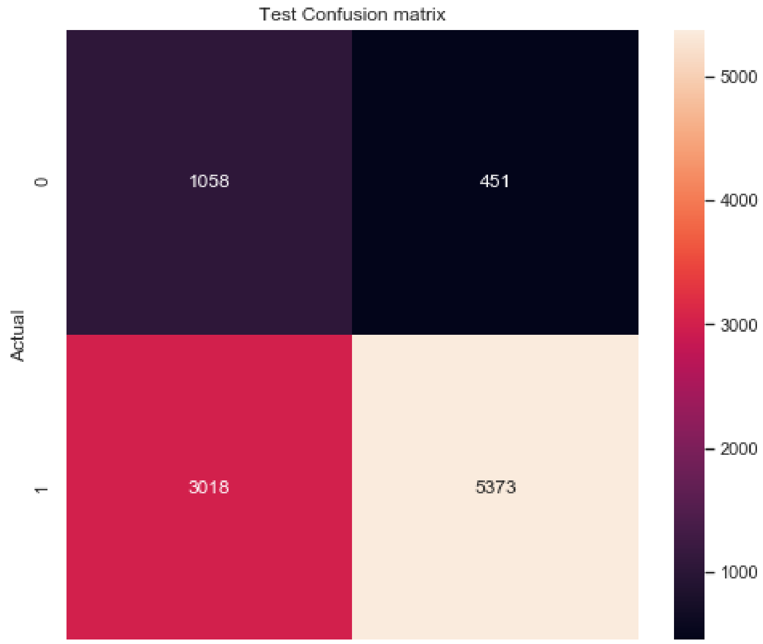
In [568]:

```
print("Test confusion matrix")
cm1=confusion_matrix(y_test_1, y_predict_thres)
print(cm1)


#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")
```

```
Test confusion matrix
[[1058  451]
 [3018 5373]]
```

Out[568]:

```
Text(0.5, 39.5, 'Predicted')
```



Set 4: Categorical Features,Numerical Features+Preprocessed Essay(tf-idf W2Vec)+Preprocessed Title(tf-idf W2Vec)

In [569]:

```
from scipy.sparse import hstack
X_tr_44=hstack((train_coded_cat ,train_coded_subcat ,train_coded_prefix, train_coded_grade
,train_coded_state,tfidf_w2v_vectors_train, tfidf_w2v_vectors_title_train,price_train_1,
quantity_train_1 ,tnp_train_1)).tocsr()

X_cv_44=hstack((cv_coded_cat ,cv_coded_subcat ,cv_coded_prefix, cv_coded_grade
,cv_coded_state,tfidf_w2v_vectors_cv, tfidf_w2v_vectors_title_cv,price_cv_1 ,quantity_cv_1,
tnp_cv_1)).tocsr()

X_te_44=hstack((test_coded_cat ,test_coded_subcat ,test_coded_prefix, test_coded_grade
,test_coded_state,tfidf_w2v_vectors_test, tfidf_w2v_vectors_title_test,price_test_1,
quantity_test_1 ,tnp_test_1)).tocsr()
```

In [570]:

```
#checking the final matrix are of same dimension or not
print(X_tr_44.shape,y_train_1.shape)
print("="*50)
print(X_cv_44.shape,y_cv_1.shape)
print("="*50)
print(X_te_44.shape,y_test_1.shape)
```

```
(13467, 613) (13467,)
==================================================
(6633, 613) (6633,)
==================================================
(9900, 613) (9900,)
```

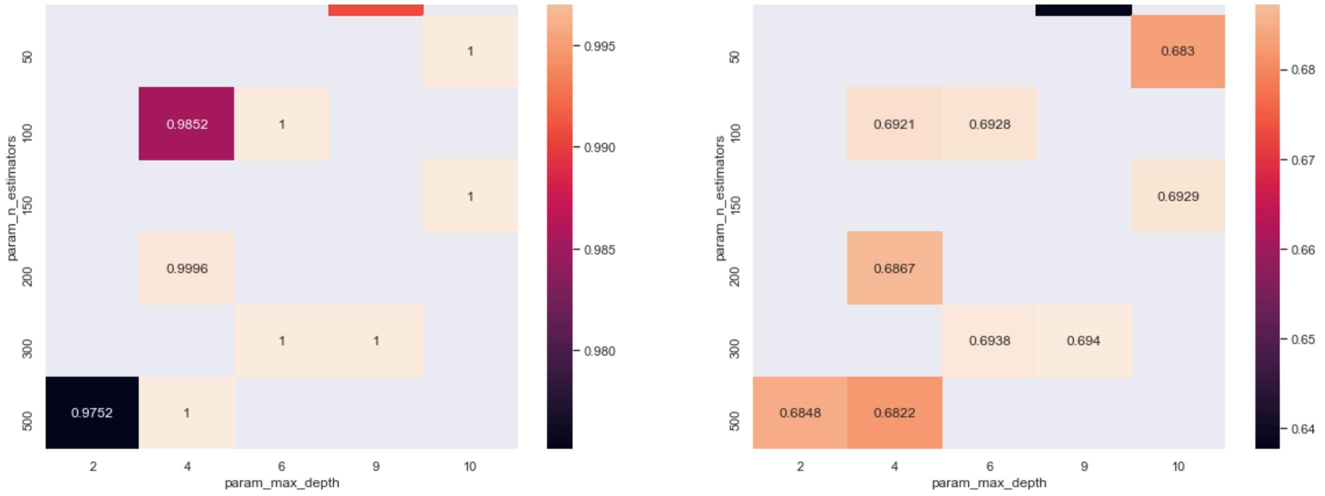
finding best Hyperparameters using RandomizedSearchCV

In [571]:

```
import xgboost as xgb
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import cross_val_score
clf_xgb = xgb.XGBClassifier()
parameters={'n_estimators':[10, 50, 100, 150, 200, 300, 500], 'max_depth':[2, 3, 4, 5, 6, 7, 8, 9, 1
0]}

clf=RandomizedSearchCV(clf_xgb,parameters, cv=2, scoring='roc_auc', return_train_score=True)

set44=clf.fit(X_tr_44,y_train_1)
```

In [572]:

```
import seaborn as sns
sns.set()
df44=pd.DataFrame(clf.cv_results_).groupby(['param_n_estimators',
'param_max_depth']).max().unstack()[['mean_test_score', 'mean_train_score']]
fig,ax=plt.subplots(1,2, figsize=(20,8))
sns.heatmap(df44.mean_train_score,annot=True, fmt='.4g', ax=ax[0])
sns.heatmap(df44.mean_test_score,annot=True, fmt='.4g', ax=ax[1])
ax[0].set_title("Train_Set")
ax[1].set_title("CV_Set")
plt.show()
```



In [573]:

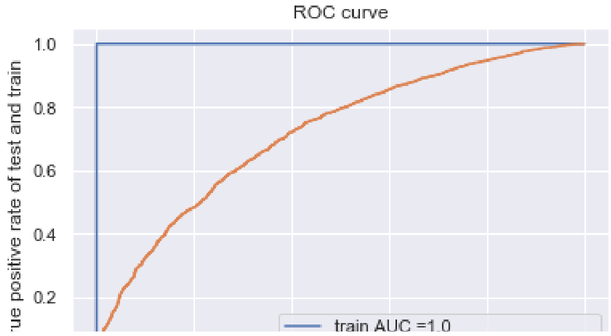
```
print(clf.best_estimator_)

print(clf.score(X_tr_44,y_train_1))
print(clf.score(X_cv_44,y_cv_1))
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=0.1,
       max_delta_step=0, max_depth=9, min_child_weight=1, missing=None,
       n_estimators=300, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
1.0
0.7192840625311216
```

Testing on Test Data(using our max_depth=9 and n_estimators=300)

In [574]:

```
clf_xgb = xgb.XGBClassifier(n_estimators=300, max_depth=9,class_weight='balanced',learning_rate=0.1
)

clf_xgb.fit(X_tr_44, y_train_1)
train_predict=clf_xgb.predict_proba(X_tr_44)[:,1]
test_predict= clf_xgb.predict_proba(X_te_44)[:,1]
train_fpr,train_tpr,train_thresholds= roc_curve(y_train_1,train_predict)
test_fpr,test_tpr,test_thresholds= roc_curve(y_test_1,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(True)
plt.show()
```



Tr
0.0
test AUC =0.7181832928855975
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate of test and train

Confusion Matrix

In [575]:

```
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   1.999950
1  0.0  0.000088   0.999950
2  0.0  0.001843   0.999908
(1939, 3)
```

In [576]:

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

In [577]:

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

In [578]:

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

df.head(3)
```

Out[578]:

	fpr	tpr	threshold	Specificty	Value
1937	0.0	1.000000	0.978916	1.0	1.000000
1936	0.0	0.996490	0.987090	1.0	0.996490
1935	0.0	0.996314	0.987112	1.0	0.996314

In [579]:

```
index = df.Value.argmax()
```

In [580]:

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

```
0.97891587
```

In [581]:

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

```
[1.]
```

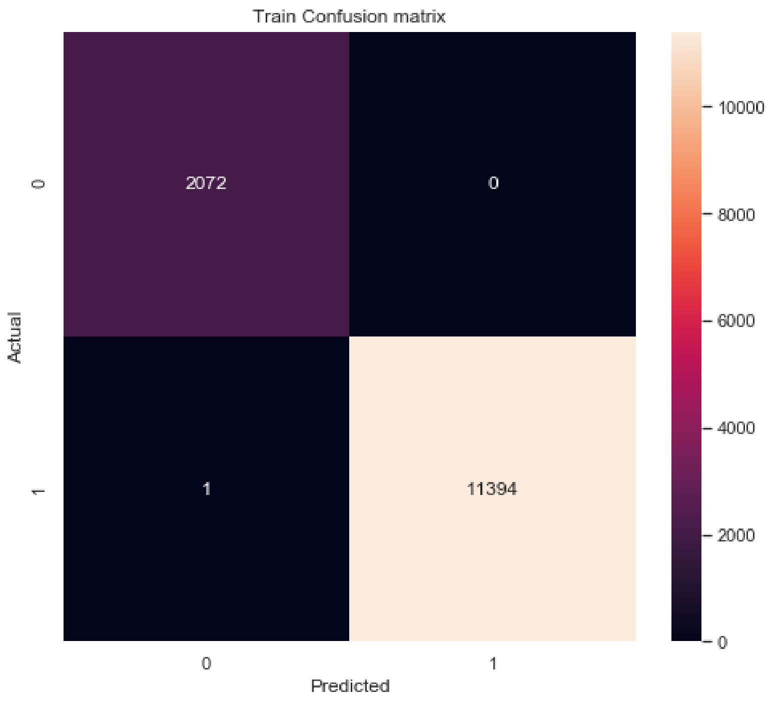
In [582]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("confusion matrix")
cm=confusion_matrix(y_train_1, y_predict_thres)
print(cm)

#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")
```

```
Threshold 0.97891587
confusion matrix
[[ 2072     0]
 [    1 11394]]
```

Out[582]:

```
Text(0.5, 39.5, 'Predicted')
```



Test Data

In [584]:

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

```
[1.]
```

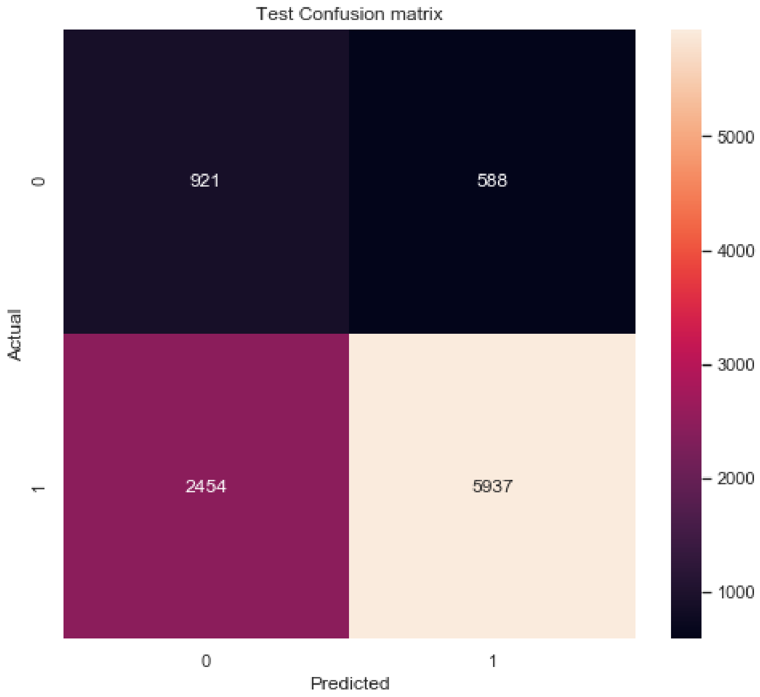
In [585]:

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


print("Test confusion matrix")
cm1=confusion_matrix(y_test_1, y_predict_thres)
print(cm1)
print(cm1)


#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")
```

```
Threshold 0.97891587
Test confusion matrix
[[ 921  588]
 [2454 5937]]
```

Out[585]:

```
Text(0.5, 39.5, 'Predicted')
```



Summarizing using Pretty Table

Random Forest

In [588]:

```
#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","Model","Search-Param","Best Hyperparameter","Test AUC"] #column headers

x.add_row(["I","Random Forest" ,"Random Search", "max-depth=8,n_estimators=1000" , 0.712])
x.add_row(["II", "Random Forest","Random Search", "max-depth=9,n_estimators=500" , 0.642])
x.add_row(["III","Random Forest","Random Search" ,"max-depth=6,n_estimators=1000", 0.709])
x.add_row(["IV","Random Forest","Random Search", "max-depth=6,n_estimators=500" , 0.703])
print(x)
```

```
+-----+---------------+---------------+-------------------------------+----------+
| SET |     Model     |  Search-Param |      Best Hyperparameter      | Test AUC |
+-----+---------------+---------------+-------------------------------+----------+
|  I  | Random Forest | Random Search | max-depth=8,n_estimators=1000 |  0.712   |
|  II | Random Forest | Random Search |  max-depth=9,n_estimators=500 |  0.642   |
| III | Random Forest | Random Search | max-depth=6,n_estimators=1000 |  0.709   |
|  IV | Random Forest | Random Search |  max-depth=6,n_estimators=500 |  0.703   |
+-----+---------------+---------------+-------------------------------+----------+
```

XG Boost

In [589]:

```
#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","Model","Search-Param","Best Hyperparameter","Test AUC"] #column headers

x.add_row(["I","XG-Boost" ,"Random Search", "max-depth=2,n_estimators=150" , 0.733])
x.add_row(["II", "XG-Boost","Random Search", "max-depth=2,n_estimators=200" , 0.737])
x.add_row(["III","XG-Boost","Random Search" ,"max-depth=10,n_estimators=500", 0.729])
x.add_row(["IV","XG-Boost","Random Search", "max-depth=9,n_estimators=300" , 0.718])
print(x)
```

```
+-----+----------+---------------+-------------------------------+----------+
| SET |  Model   |  Search-Param |      Best Hyperparameter      | Test AUC |
+-----+----------+---------------+-------------------------------+----------+
|  I  | XG-Boost | Random Search |  max-depth=2,n_estimators=150 |  0.733   |
|  II | XG-Boost | Random Search |  max-depth=2,n_estimators=200 |  0.737   |
| III | XG-Boost | Random Search | max-depth=10,n_estimators=500 |  0.729   |
|  IV | XG-Boost | Random Search |  max-depth=9,n_estimators=300 |  0.718   |
+-----+----------+---------------+-------------------------------+----------+
```

In [ ]: