1. 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.

2. 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, opening 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; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

3.Directory List

In [2]:

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

4. About the dataset

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

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

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

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

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

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

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

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

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

Notes on the Essay Data

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

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_4:__ "Close by sharing why your project will make a difference"

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

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

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

5. Reading the data

In [3]:

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

In [4]:

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

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

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

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

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

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

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

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

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

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

In [5]:

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

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


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


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

Out[5]:

Out[5]:

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

In [6]:

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

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

    price
0  149.00
1   14.95
2    8.45
```

In [7]:

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

price_data = res_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index() #grouping
is done on the basis of ids and agggreating 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 [8]:

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

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

Imbalanced Dataset where class-label 1 is 85% and 0 is 15%

Feature Preprocessing

Preprocessing of project_subject_categories

In [9]:

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

```
mathscience
```

In [10]:

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

In [11]:

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

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

```
print(sortd)

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

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

Preprocessing of project_subject_subcategories

In [12]:

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

```
specialneeds
```

In [13]:

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

In [14]:

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

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

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

```
{'appliedsciences': 10816, 'healthlifescience': 4235, 'specialneeds': 13642, 'literacy': 33700, 'earlydevelopment': 4254, 'mathematics': 28074, 'socialsciences': 1920, 'historygeography': 3171, 'e
```

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

Text Preprocessing

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

In [15]:

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

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

Essay Text

In [16]:

```
# 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
```

\" 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 [17]:

```
# 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"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [18]:

```
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 [19]:

```
# \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 [20]:

```
#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
```

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 [21]:

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

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

In [22]:

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

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[00:55<00:00, 1978.68it/s]
```

In [23]:

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

Out[23]:

```
'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 [24]:

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

Project title text

In [25]:

```
# 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 [26]:

```
#1.Decontraction
test1 = decontracted(train_data['project_title'].values[7])
print(test1)
print("="*50)
```

```
21st Century Learning with Multimedia
==================================================
```

In [27]:

```
#2. Removing newline breakline etc
# \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)
```

```
21st Century Learning with Multimedia
```

In [28]:

```
#remove spacial 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)
```

```
21st Century Learning with Multimedia
```

In [29]:

In [29]:

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

```
100%|████████████████████████████████████████████████████████████| 109248/109248
[00:05<00:00, 19536.61it/s]
```

In [30]:

```
preprocessed_title[0]
```

Out[30]:

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

In [31]:

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

Category Preprocessing

Teacher Prefix

In [32]:

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

Out[32]:

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

Need to convert it into lowercase as well as remove the punctuation at the last

In [33]:

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

```
100%|████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 349122.14it/s]
```

In [34]:

```
preprocessed_prefix[3]
```

Out[34]:

```
'mrs'
```

In [35]:

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

Grade Category

In [36]:

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

Out[36]:

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

In [37]:

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

Out[37]:

```
Grades PreK-2    44225
Grades 3-5       37137
Grades 6-8       16923
Grades 9-12      10963
Name: project_grade_category, dtype: int64
```

In [38]:

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

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248
[00:00<00:00, 320322.44it/s]
```

In [39]:

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

Out[39]:

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

In [40]:

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

project_resource_summary

In [41]:

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

Out[41]:

```
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 [42]:

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

```
100%|████████████████████████████████████████████████████████████| 109248/109248
[00:09<00:00, 12082.03it/s]
```

In [43]:

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

Out[43]:

```
['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 [44]:

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

In [45]:

```
#how to drop a column in pandas-> https://pandas.pydata.org/pandas-
docs/stable/reference/api/pandas.DataFrame.drop.html
print(train_data.shape)
```

```
(109248, 21)
```

In [46]:

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

In [47]:

```
print(x.head(3))
```

```
   Unnamed: 0 teacher_prefix school_state  \
0        8393           Mrs.           CA
1       37728            Ms.           UT
2       74477           Mrs.           CA

   teacher_number_of_previously_posted_projects  project_is_approved   price  \
0                                            53                   1  725.05
1                                             4                   1  213.03
2                                            10                   1  329.00

   quantity  clean_categories               clean_subcategories  \
0         4       mathscience  appliedsciences healthlifescience
1         8      specialneeds                       specialneeds
2         1  literacylanguage                           literacy

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

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

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

In [48]:

```
y1=x['project_is_approved']
```

In [49]:

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

In [50]:

```
print(x.head(3))
```

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

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

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

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

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

In [51]:

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

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

Data Splitting into train,cv and test

In [52]:

```
# ============================== 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.neighbors import KNeighborsClassifier
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 [53]:

```
# split the data set into train and test
#how to stratify using knn->https://stackoverflow.com/questions/34842405/parameter-stratify-from-method-train-test-split-scikit-learn
X_1, X_test, y_1, y_test =model_selection.train_test_split(x,y1, test_size=0.33, random_state=5,stratify= y1)#random spliiting of data into test and train
```

In [54]:

```
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 [55]:

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

print("="*100)
```

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

In [56]:

```
print(X_train.head(3))
```

```
        Unnamed: 0 school_state  teacher_number_of_previously_posted_projects  \
57779         95236           GA                                            0
71338        157264           TX                                            6
109035       101741           HI                                            7

         price  quantity   clean_categories clean_subcategories  \
57779    49.99        15   literacylanguage   literaturewriting
71338    32.00       107   literacylanguage   literaturewriting
109035  246.91        37  warmth carehunger   warmth carehunger

                                    preprocessed_essays  \
57779   students placed small group reading class mean...
71338   students come predominately low income familie...
109035  school located low socioeconomic area 14 stude...

                                    preprocessed_title preprocessed_prefix  \
```

```
57779                                    fire learning                  mr
71338                                         supplies                 mrs
109035  meet parents parental engagement snacks supplies                  ms

       preprocessed_grade                                preprocessed_resource
57779         Grades_9_12  students need 15 amazon fire tablets access eb...
71338          Grades_3_5  students need basic school supplies help succe...
109035         Grades_3_5  students need art supplies tote bags chapter b...
```

Vectorization

One-Hot encoding of categorical feature

Category Feature

In [57]:

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

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

Sub-Category feature

In [58]:

```
vectorizer_sub_cat = CountVectorizer(vocabulary=list(sorted_subcat_dict.keys()), lowercase=False,
binary=True)
```

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

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

School-State feature

In [59]:

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

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

In [60]:

```
vectorizer_school = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lowercase=Fa
lse, binary=True)
vectorizer_school.fit(X_train['school_state'].values)
print(vectorizer_school.get_feature_names())
print('='*50)
```

```
state_ohe_train=vectorizer_school.transform(X_train['school_state'].values)#applying learned
parameters to train,test and cv values
print("Shape of train data after one hot encoding",state_ohe_train.shape)
print("train data after one hot encoding",state_ohe_train[0:5,:])
state_ohe_cv=vectorizer_school.transform(X_cv['school_state'].values)
print('='*50)
print("Shape of CV data after one hot encoding",state_ohe_cv.shape)
print("CV data after one hot encoding",state_ohe_cv[0:5,:])
state_ohe_test=vectorizer_school.transform(X_test['school_state'].values)
print('='*50)
print("Shape of test data after one hot encoding",state_ohe_test.shape)
print("test data after one hot encoding",state_ohe_test[0:5,:])
```

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

Project_Grade feature

In [61]:

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

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

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

In [62]:

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

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

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

In [63]:

```
vectorizer_grade = CountVectorizer(vocabulary=list(sorted_project_grade_dict.keys()), lowercase=Fa
```

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

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

Teacher-Prefix feature

In [64]:

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

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

In [65]:

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

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

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

In [66]:

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

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

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

Normalizing Numerical Features

Price feature

In [67]:

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


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

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

In [69]:

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

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

Difference between reshape(1,-1) and reshape(-1,1)

https://stackoverflow.com/questions/18691084/what-does-1-mean-in-numpy-reshape/42950520

Quantity Feature

In [72]:

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


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

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

In [73]:

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

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

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

Teacher number of previously posted projects feature

In [74]:

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

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

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

In [75]:

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

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

Vectorizing Test Data

1. Bag of words(BoW)

Preprocessed Essay

In [76]:

```
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  (49041, 12015)
==================================================
Shape of matrix  (24155, 12015)
==================================================
Shape of matrix  (36052, 12015)
```

Preprocessed Title

In [77]:

```
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  (49041, 2010)
==================================================
Shape of matrix  (24155, 2010)
==================================================
Shape of matrix  (36052, 2010)
```

2. Tf-idf

Preprocessed Essay

In [78]:

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

```
Shape of matrix  (49041, 12015)
==================================================
Shape of matrix  (24155, 12015)
==================================================
Shape of matrix  (36052, 12015)
```

Preprocessed Title

In [79]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
model_title_tfidf = TfidfVectorizer(min_df=10) #df tells us that we will only consider those words
which is present atleast in 10 documents
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"]) #BoW of cv
print("Shape of matrix ",cv_tfidf_title.shape)
print("="*50)
test_tfidf_title= model_title_tfidf.transform(X_test["preprocessed_title"]) #BoW of test
print("Shape of matrix ",test_tfidf_title.shape)
```

```
Shape of matrix  (49041, 2010)
==================================================
Shape of matrix  (24155, 2010)
==================================================
Shape of matrix  (36052, 2010)
```

Applying Naive Bayes

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

In [80]:

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

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

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

In [81]:

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

```
(49041, 14128) (49041,)
==================================================
(24155, 14128) (24155,)
==================================================
(36052, 14128) (36052,)
```

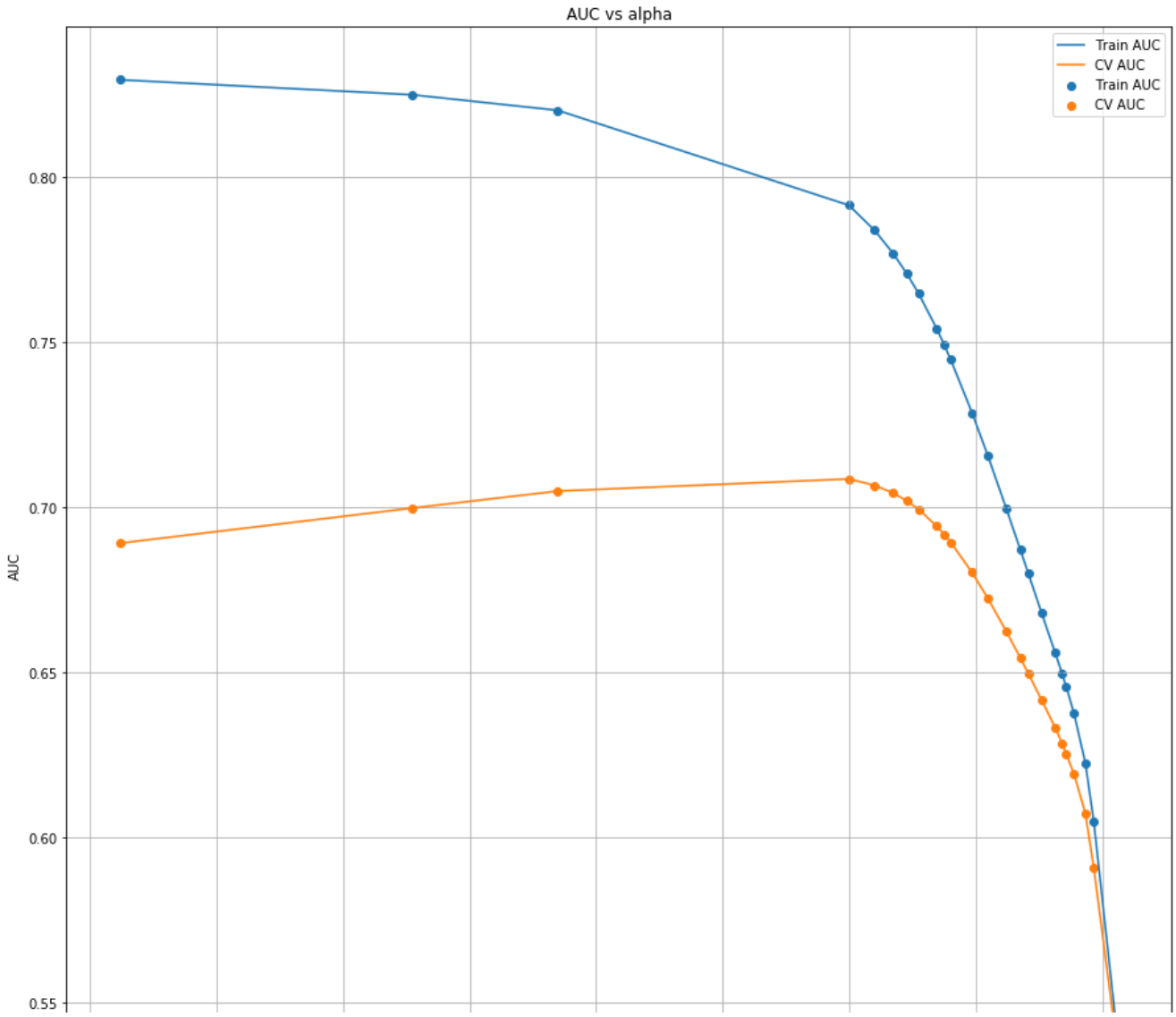
Simple Brute Force

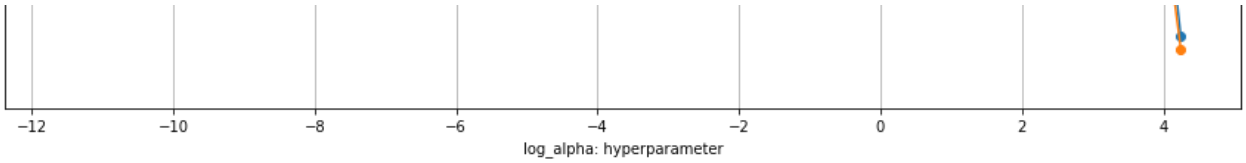
Finding Hyper(alpha) parameter(alpha)using AUC value

In [82]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook as tqdm
import math
alpha1=[]
train_auc = []
cv_auc = []
alpha = [ 0.00001  0.001  0.01  1  1.5  2  2.5  3.001  4  4.5  5  7  9  12  15  17  21  26  29  31  35
```

```
alpha = [ 0.00001, 0.001, 0.01,1 ,1.5 ,2 ,2.5 ,3.001 ,4, 4.5 ,5,7,9, 12, 15 ,17, 21 ,26, 29, 31 ,35
,42 ,48,70]
for i in tqdm(alpha):
    neigh = MultinomialNB(alpha=i, class_prior=[0.5,0.5]) #to deal with class imbalance we used cla
ss_prior as 0.5,0.5
    neigh.fit(X_tr, y_train) #during fit our model is learning from the training data e.g. y=f(x)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    y_train_pred =  neigh.predict_log_proba(X_tr)[:,1]
    y_cv_pred =  neigh.predict_log_proba(X_cv)[:,1]
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pre
diction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

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

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

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

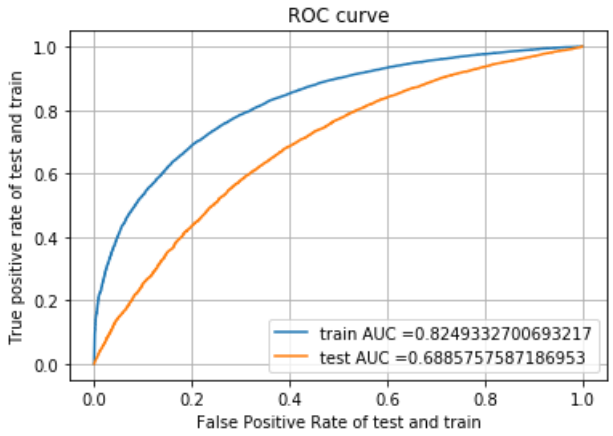




Testing on Test Data(using our best hyper parameter=1)

In [83]:

```
from sklearn.metrics import roc_curve,auc

neigh=MultinomialNB(alpha=0.001)
neigh.fit(X_tr,y_train)

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

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

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



Confusion Matrix

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

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

Train Data

In [84]:

In [84]:

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

```
        fpr       tpr     threshold
0  0.000000  0.000000  1.000000e+00
1  0.000404  0.022564  0.000000e+00
2  0.000404  0.023165 -1.136868e-13
(10467, 3)
```

In [85]:

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

In [86]:

```
df.head(3)
```

Out[86]:

	fpr	tpr	threshold	Specificty
0	0.000000	0.000000	1.000000e+00	1.000000
1	0.000404	0.022564	0.000000e+00	0.999596
2	0.000404	0.023165	-1.136868e-13	0.999596

In [87]:

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

In [88]:

```
df.head(3)
```

Out[88]:

	fpr	tpr	threshold	Specificty	Value
0	0.000000	0.000000	1.000000e+00	1.000000	0.000000
1	0.000404	0.022564	0.000000e+00	0.999596	0.022555
2	0.000404	0.023165	-1.136868e-13	0.999596	0.023155

In [89]:

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

In [90]:

```
df.head(3)
```

Out[90]:

	fpr	tpr	threshold	Specificty	Value
3891	0.253838	0.745645	-0.131778	0.746162	0.556372
3739	0.241584	0.733558	-0.104133	0.758416	0.556342
3923	0.256396	0.748144	-0.138910	0.743604	0.556322

In [91]:

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

In [92]:

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

```
-0.13177794517503116
```

In [93]:

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

```
[1.]
```

In [94]:

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

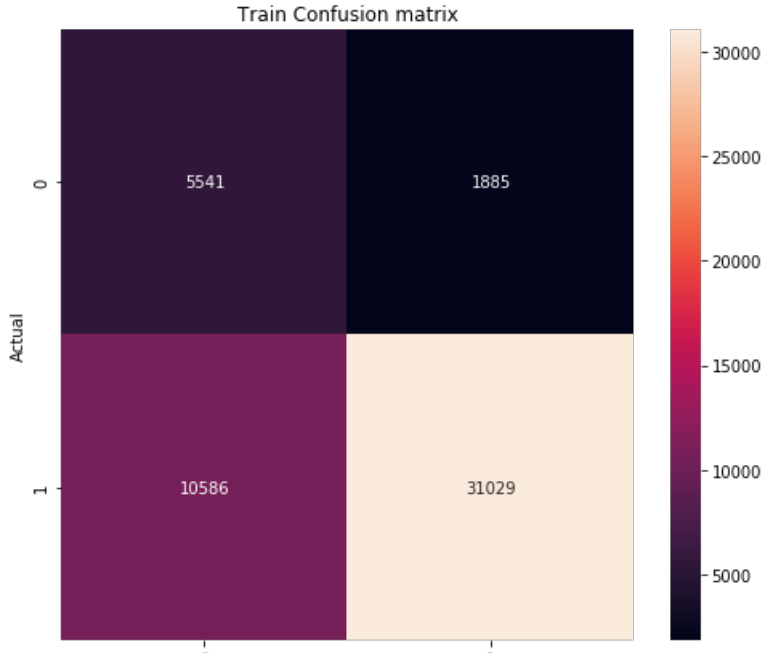
```
Threshold -0.13177794517503116
Train confusion matrix
[[ 5541  1885]
 [10586 31029]]
```

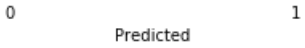
In [95]:

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

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

Out[95]:

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





Test Data

In [96]:

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

```
[1.]
```

In [97]:

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

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

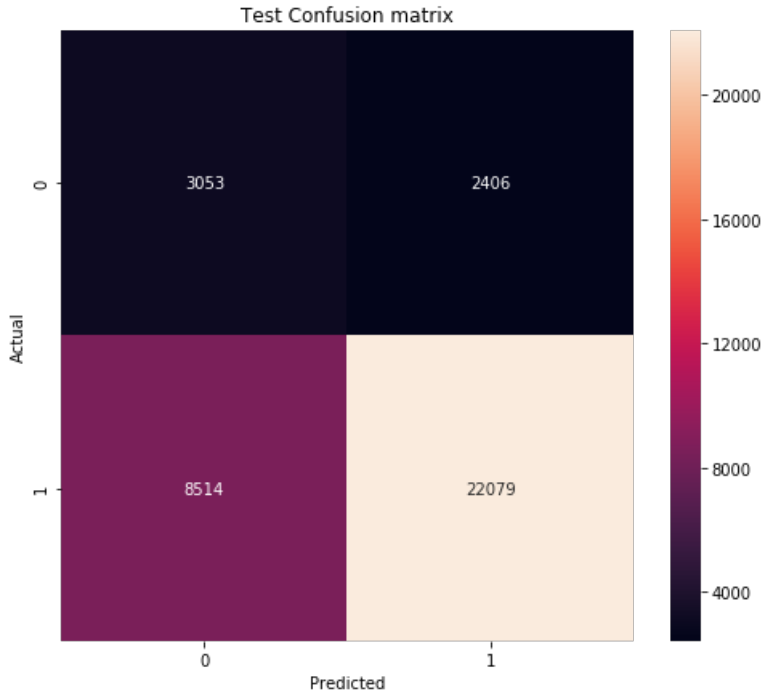
```
Threshold -0.13177794517503116
Test confusion matrix
[[ 3053  2406]
 [ 8514 22079]]
```

In [98]:

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

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

Out[98]:

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



In [99]:

```
features=[]
for a in vectorizer_cat.get_feature_names():
    features.append(a)

len(features)
```

Out[99]:

9

In [100]:

```
for a in vectorizer_sub_cat.get_feature_names():
    features.append(a)

len(features)
```

Out[100]:

39

In [101]:

```
for a in  vectorizer_school.get_feature_names():
    features.append(a)

len(features)
```

Out[101]:

90

In [102]:

```
for a in  vectorizer_grade.get_feature_names():
    features.append(a)

len(features)
```

Out[102]:

94

In [103]:

```
for a in  vectorizer_prefix.get_feature_names():
    features.append(a)

len(features)
```

Out[103]:

100

In [104]:

```
features.append("price")
len(features)
```

Out[104]:

101

In [105]:

```
features.append("quantity")
features.append("teacher_number_of_previously_posted_projects")
```

```
len(features)
```

Out[105]:

```
103
```

In [106]:

```
for a in    model_essay_bow.get_feature_names():
        features.append(a)

len(features)
```

Out[106]:

```
12118
```

In [107]:

```
for a in    model_title_bow.get_feature_names():
        features.append(a)

len(features)
```

Out[107]:

```
14128
```

In [108]:

```
#how to get top feature names in naive bayes algorithm ->
https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes


neg_class_prob_bow = neigh.feature_log_prob_[0, :]
pos_class_prob_bow = neigh.feature_log_prob_[1, :]
```

In [109]:

```
print("shape of positive",pos_class_prob_bow.shape)
print(type(pos_class_prob_bow))
print("="*50)
print("shape of negative",neg_class_prob_bow.shape)
print(type(neg_class_prob_bow))
```

```
shape of positive (14128,)
<class 'numpy.ndarray'>
==================================================
shape of negative (14128,)
<class 'numpy.ndarray'>
```

In [110]:

```
positive_features=pd.DataFrame({"features" : features , "positive_probabilities" :
pos_class_prob_bow})
```

In [111]:

```
positive_features.head(3)
```

Out[111]:

	features	positive_probabilities
0	carehunger	-9.281084
1	warmth	-9.281084
2	historycivics	-7.881071

In [112]:

```
negative_features=pd.DataFrame({"features" : features , "negative_probabilities" :
neg_class_prob_bow})
negative_features.head(3)
```

Out[112]:

	features	negative_probabilities
0	carehunger	-9.861221
1	warmth	-9.861221
2	historycivics	-7.931632

In [113]:

```
top_20=positive_features.sort_values(by=["positive_probabilities"], ascending=False)
top_20.head(20)
```

Out[113]:

	features	positive_probabilities
10474	students	-2.988929
9509	school	-4.130002
6314	learning	-4.498165
2130	classroom	-4.525025
6310	learn	-4.841417
5223	help	-4.861107
6681	many	-5.007469
7177	nannan	-5.019637
8763	reading	-5.134449
7227	need	-5.135972
11992	work	-5.143569
11532	use	-5.188853
6552	love	-5.296468
2931	day	-5.317678
314	able	-5.323793
2267	come	-5.347479
2117	class	-5.365859
12033	would	-5.400312
10796	technology	-5.460186
1451	books	-5.472010

In [114]:

```
top_20_negative=negative_features.sort_values(by=["negative_probabilities"], ascending=False)
top_20_negative.head(20)
```

Out[114]:

	features	negative_probabilities
10474	students	-2.994975
9509	school	-4.094571
6314	learning	-4.400130
2130	classroom	-4.554468

	features	negative_probabilities
6310	learn	-4.755752
5223	help	-4.788191
7177	nannan	-4.967209
6681	many	-5.012014
7227	need	-5.087841
11992	work	-5.127693
2267	come	-5.308357
6552	love	-5.336993
8763	reading	-5.346449
6739	materials	-5.357246
2931	day	-5.370063
314	able	-5.370909
9903	skills	-5.374301
11532	use	-5.413496
2117	class	-5.420588
11754	want	-5.443992

In [115]:

```
a=neigh.coef_[0,:]
print(a.shape)
```

```
(14128,)
```

In [116]:

```
top_bow=pd.DataFrame({'features' : features, "probabilities" : a})
```

In [117]:

```
top_bow=top_bow.sort_values(by=["probabilities"], ascending=False)
```

In [118]:

```
top_bow.head(20)
```

Out[118]:

	features	probabilities
10474	students	-2.988929
9509	school	-4.130002
6314	learning	-4.498165
2130	classroom	-4.525025
6310	learn	-4.841417
5223	help	-4.861107
6681	many	-5.007469
7177	nannan	-5.019637
8763	reading	-5.134449
7227	need	-5.135972
11992	work	-5.143569
11532	use	-5.188853
6552	love	-5.296468
2931	day	-5.317678
314	able	-5.323793
2267	come	-5.347479

	features	probabilities
2117	class	-5.365859
12033	would	-5.400312
10796	technology	-5.460186
1451	books	-5.472010

Set 2: Categorical Features,Numerical Features+Preprocessed Essay(Tfidf)+Preprocessed Title(Tfidf)

In [119]:

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

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

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

In [120]:

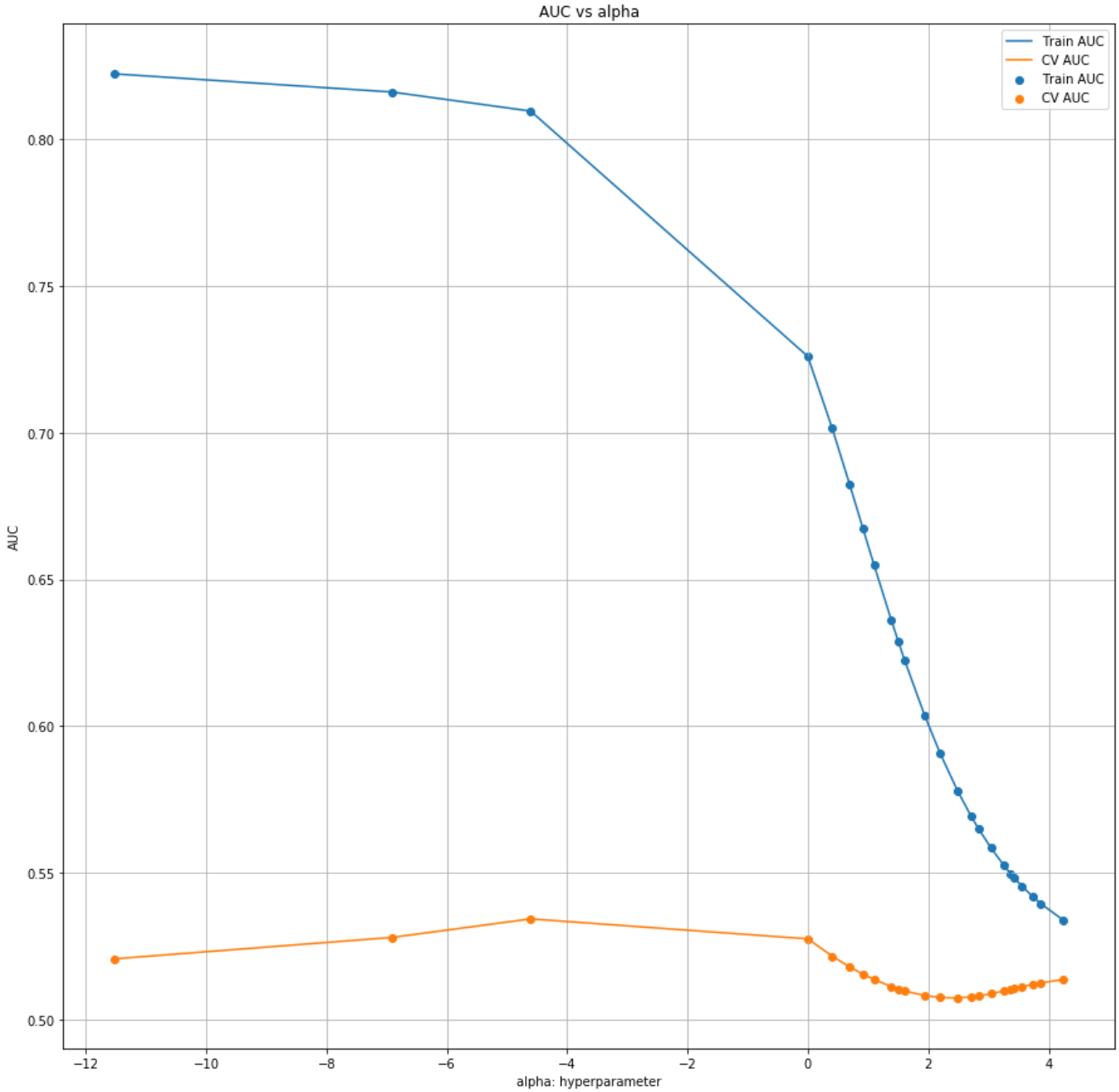
```
#checking the final matrix are of same dimension or not
print(X_tr.shape,y_train.shape)
print("="*50)
print(X_cv.shape,y_cv.shape)
print("="*50)
print(X_te.shape,y_test.shape)
```

```
(49041, 14128) (49041,)
==================================================
(24155, 14128) (24155,)
==================================================
(36052, 14128) (36052,)
```

In [121]:

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from tqdm import tqdm_notebook as tqdm
train_auc = []
cv_auc = []
alpha1=[]
alpha = [0.00001, 0.001, 0.01,1 ,1.5 ,2 ,2.5 ,3.001 ,4, 4.5 ,5,7,9, 12, 15 ,17, 21 ,26, 29, 31 ,35 ,
42 ,48,70]
for i in tqdm(alpha):
    neigh = MultinomialNB(alpha=i)
    neigh.fit(X_tr, y_train) #during fit our model is learning from the training data e.g. y=f(x)
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    y_train_pred =  neigh.predict_log_proba(X_tr)[:,1]
    y_cv_pred =  neigh.predict_log_proba(X_cv)[:,1]
    train_auc.append(roc_auc_score(y_train,y_train_pred))#roc_auc_score->Compute(ROC AUC) from pre
diction scores.
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))

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

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

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

plt.legend() #adding legend
```

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

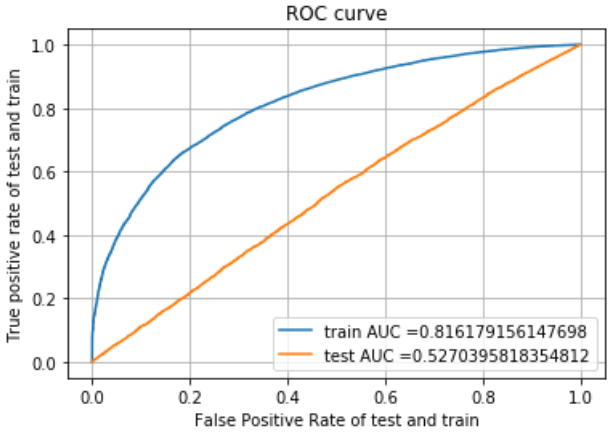


Testing on Test Data(using our best hyper parameter=0.001)

In [122]:

```
from sklearn.metrics import roc_curve,auc

neigh=MultinomialNB(alpha=0.001)
neigh.fit(X_tr,y_train)

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

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

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



Confusion Matrix

Train Data

In [123]:

```
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  9.999998e-01
1  0.0  0.000024 -2.021333e-07
2  0.0  0.059041 -3.319335e-03
(10168, 3)
```

In [124]:

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

In [125]:

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

In [126]:

```
df.head(3)
```

Out[126]:

	fpr	tpr	threshold	Specificty	Value
0	0.0	0.000000	9.999998e-01	1.0	0.000000
1	0.0	0.000024	-2.021333e-07	1.0	0.000024
2	0.0	0.059041	-3.319335e-03	1.0	0.059041

In [127]:

In [127]:

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

Out[127]:

	fpr	tpr	threshold	Specificty	Value
3596	0.266361	0.741199	-0.164701	0.733639	0.543772
3600	0.266631	0.741463	-0.164849	0.733369	0.543766
3602	0.266765	0.741536	-0.164874	0.733235	0.543719

In [128]:

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

```
-0.16470147624232823
```

In [129]:

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

```
[1.]
```

In [130]:

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

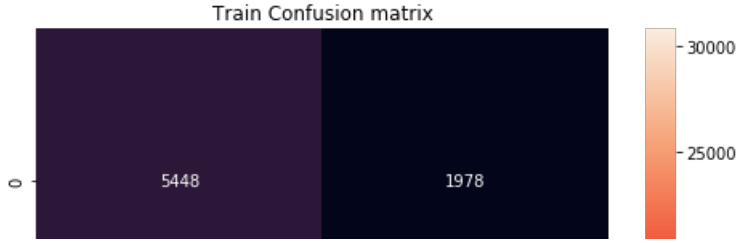
```
Threshold= -0.16470147624232823
Train confusion matrix
[[ 5448  1978]
 [10771 30844]]
```

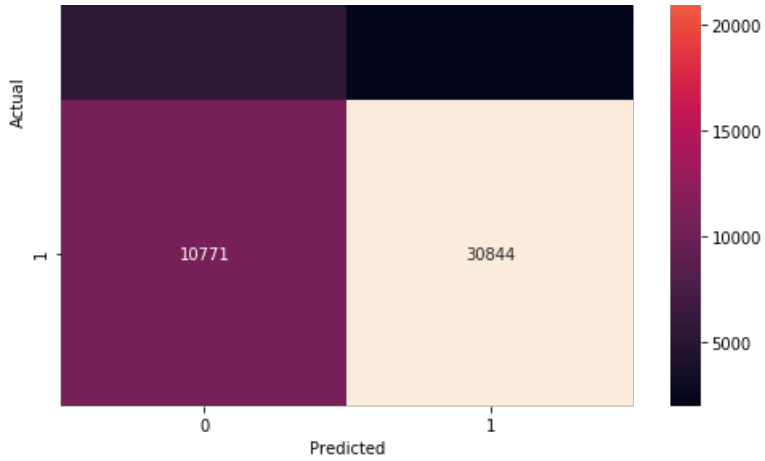
In [131]:

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

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

Out[131]:

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





Test Data

In [132]:

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

```
[1.]
```

In [133]:

```
from sklearn.metrics import confusion_matrix
print("Threshold",a)
print("Test confusion matrix")
cm1=confusion_matrix(y_test,y_predict_thres)
print(cm1)
```

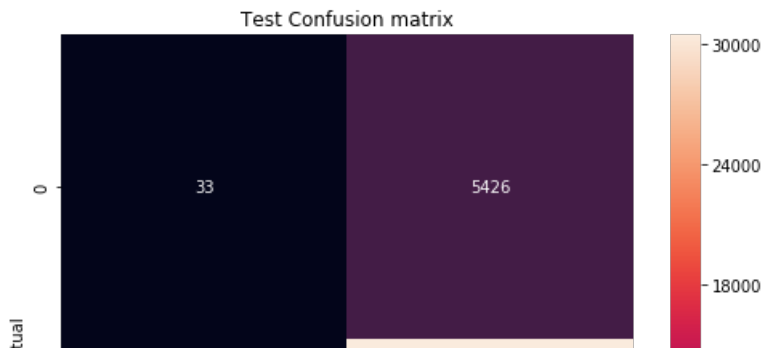
```
Threshold -0.16470147624232823
Test confusion matrix
[[   33  5426]
 [  133 30460]]
```

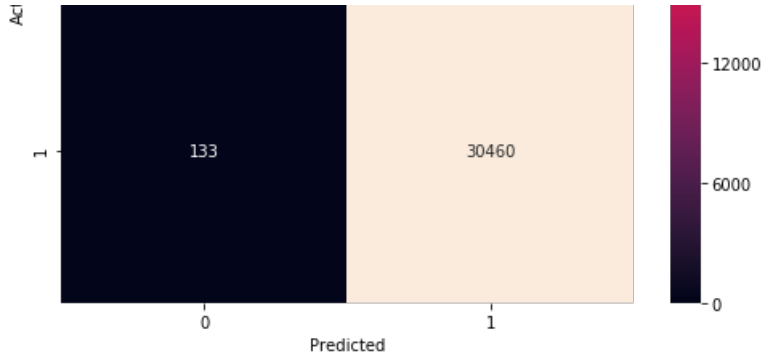
In [134]:

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

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

Out[134]:

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





Top 20 Features of the Set 2

In [135]:

```
features_tfidf=[]
for a in vectorizer_cat.get_feature_names():
    features_tfidf.append(a)

len(features_tfidf)
```

Out[135]:

9

In [136]:

```
for a in vectorizer_sub_cat.get_feature_names():
    features_tfidf.append(a)

len(features_tfidf)
```

Out[136]:

39

In [137]:

```
for a in  vectorizer_school.get_feature_names():
    features_tfidf.append(a)

len(features_tfidf)
```

Out[137]:

90

In [138]:

```
for a in  vectorizer_grade.get_feature_names():
    features_tfidf.append(a)

len(features_tfidf)
```

Out[138]:

94

In [139]:

```
for a in  vectorizer_prefix.get_feature_names():
    features_tfidf.append(a)

len(features_tfidf)
```

Out[139]:

100

In [140]:

```
features_tfidf.append("price")
len(features_tfidf)
```

Out[140]:

101

In [141]:

```
features_tfidf.append("quantity")
len(features_tfidf)
```

Out[141]:

102

In [142]:

```
features_tfidf.append("teacher_number_of_previously_posted_projects")
len(features_tfidf)
```

Out[142]:

103

In [143]:

```
for a in model_essay_tfidf.get_feature_names():
        features_tfidf.append(a)

len(features_tfidf)
```

Out[143]:

12118

In [144]:

```
for a in model_title_tfidf.get_feature_names():
        features_tfidf.append(a)

len(features_tfidf)
```

Out[144]:

14128

In [145]:

```
#how to get top feature names in naive bayes algorithm ->
https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes


neg_class_prob_tfidf = neigh.feature_log_prob_[0, :]
pos_class_prob_tfidf = neigh.feature_log_prob_[1, :]
```

In [146]:

```
print("shape of positive",pos_class_prob_tfidf.shape)
print(type(pos_class_prob_tfidf))
print("="*50)
print("shape of negative",neg_class_prob_tfidf.shape)
print(type(neg_class_prob_tfidf))
```

```
shape of positive (14128,)
<class 'numpy.ndarray'>
==================================================
shape of negative (14128,)
<class 'numpy.ndarray'>
```

In [147]:

```
positive_features_tfidf=pd.DataFrame({"features" : features_tfidf , "positive_probabilities" :
pos_class_prob_tfidf})
positive_features_tfidf.head(3)
```

Out[147]:

	features	positive_probabilities
0	carehunger	-7.096174
1	warmth	-7.096174
2	historycivics	-5.696161

In [148]:

```
top_20_tfidf=positive_features_tfidf.sort_values(by=["positive_probabilities"], ascending=False)
top_20_tfidf.head(20)
```

Out[148]:

	features	positive_probabilities
99	mrs	-3.425509
8	literacylanguage	-3.504324
93	Grades_PreK_2	-3.698968
7	mathscience	-3.779498
98	ms	-3.821759
92	Grades_3_5	-3.857463
38	literacy	-3.930528
37	mathematics	-4.160500
36	literaturewriting	-4.368708
91	Grades_6_8	-4.656234
89	CA	-4.738455
12484	comic	-4.813313
6	healthsports	-4.829708
5	specialneeds	-4.881867
35	specialneeds	-4.881867
4	appliedlearning	-5.002794
90	Grades_9_12	-5.104718
34	appliedsciences	-5.116466
33	healthwellness	-5.131847
97	mr	-5.153071

In [149]:

```
negative_features_tfidf=pd.DataFrame({"features" : features_tfidf , "negative_probabilities" :
neg_class_prob_tfidf})
negative_features_tfidf.head(3)
```

Out[149]:

	features	negative_probabilities

	features	negative_probabilities
0	carehunger	-7.720439
1	warmth	-7.720439
2	historycivics	-5.790849

In [150]:

```
top_20_tfidf_negative=negative_features_tfidf.sort_values(by=["negative_probabilities"], ascending
=False)
top_20_tfidf_negative.head(20)
```

Out[150]:

	features	negative_probabilities
99	mrs	-3.469539
8	literacylanguage	-3.619843
93	Grades_PreK_2	-3.677737
7	mathscience	-3.698142
98	ms	-3.772484
92	Grades_3_5	-3.905028
38	literacy	-4.100402
37	mathematics	-4.121337
36	literaturewriting	-4.452971
91	Grades_6_8	-4.617479
5	specialneeds	-4.762211
35	specialneeds	-4.762211
12484	comic	-4.805985
6	healthsports	-4.811393
89	CA	-4.816550
4	appliedlearning	-4.892808
90	Grades_9_12	-4.981444
34	appliedsciences	-5.028893
3	musicarts	-5.074683
97	mr	-5.078707

In [151]:

```
y=top_20_tfidf_negative.iloc[0:21,-1]
x=top_20_tfidf_negative.iloc[0:21,0]
```

In [152]:

```
print(x)
```

```
99                    mrs
8         literacylanguage
93           Grades_PreK_2
7              mathscience
98                      ms
92             Grades_3_5
38               literacy
37             mathematics
36       literaturewriting
91               Grades_6_8
5              specialneeds
35             specialneeds
12484                comic
6              healthsports
89                       CA
4           appliedlearning
90               Grades_9_12
```

```
90              Grades_9_12
34          appliedsciences
3                 musicarts
97                       mr
33           healthwellness
Name: features, dtype: object
```

In [153]:

```
b=neigh.coef_[0,:]
b.shape
```

Out[153]:

```
(14128,)
```

In [154]:

```
top_tfidf=pd.DataFrame({'features' : features_tfidf , "probabilities" : b})
```

In [155]:

```
top_tfidf=top_tfidf.sort_values(by=["probabilities"], ascending=False)
top_tfidf.head(20)
```

Out[155]:

	features	probabilities
99	mrs	-3.425509
8	literacylanguage	-3.504324
93	Grades_PreK_2	-3.698968
7	mathscience	-3.779498
98	ms	-3.821759
92	Grades_3_5	-3.857463
38	literacy	-3.930528
37	mathematics	-4.160500
36	literaturewriting	-4.368708
91	Grades_6_8	-4.656234
89	CA	-4.738455
12484	comic	-4.813313
6	healthsports	-4.829708
5	specialneeds	-4.881867
35	specialneeds	-4.881867
4	appliedlearning	-5.002794
90	Grades_9_12	-5.104718
34	appliedsciences	-5.116466
33	healthwellness	-5.131847
97	mr	-5.153071

Summary using PrettyTable

In [156]:

```
#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=["Vectorizer", "Model", "Best Hyperparameter","Test AUC"] #column headers

x.add_row(["BOW", "Brute", 0.001 , 0.688])
x.add_row(["BOW", "TOP 20", "-" , "-"])
x.add_row(["TFIDF", "Brute", 0.001, 0.527])
x.add_row(["TFIDF", "TOP 20", "-", "-"])

print(x)
```

```
+------------+--------+---------------------+-----------+
| Vectorizer | Model  | Best Hyperparameter | Test AUC |
+------------+--------+---------------------+-----------+
|    BOW     | Brute  |        0.001        |  0.688    |
|    BOW     | TOP 20 |          -          |    -      |
|   TFIDF    | Brute  |        0.001        |  0.527    |
|   TFIDF    | TOP 20 |          -          |    -      |
+------------+--------+---------------------+-----------+
```

In [ ]:

```
```