In [3]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
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 gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
from tqdm import tqdm
import os
```

1 PreProcessing Data

1.1 Reading Data

In [4]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

In [5]:
```
path="/content/drive/MyDrive/GBDT/GBDT Assignment/train_data.csv"
project_data=pd.read_csv(path)
```

In [6]:
```
path="/content/drive/MyDrive/GBDT/GBDT Assignment/resources.csv"
resource_data=pd.read_csv(path)
```

In [7]:
```
print("Number of data points in train data", project_data.shape)
print('_'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
__________________________________________________
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [8]:
```
print("Number of data points in resource data",resource_data.shape)
print('_'*50)
print("The attributes of data :",resource_data.columns.values)
```

```
Number of data points in resource data (1541272, 4)
__________________________________________________
The attributes of data : ['id' 'description' 'quantity' 'price']
```

1.2 Preporcessing Categorical Data

teacher_prefix

In [9]:
```
project_data['teacher_prefix'].value_counts()
```

Out[9]:
```
Mrs.       57269
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
Name: teacher_prefix, dtype: int64
```

In [10]:
```
print(project_data['teacher_prefix'].isnull().values.any())
print("Number of nan values", project_data['teacher_prefix'].isnull().values.sum())
```

```
True
Number of nan values 3
```

In [11]:
```
#replace msiing values with Mrs
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
```

In [12]:
```
project_data['teacher_prefix'].value_counts()
```

Out[12]:
```
Mrs.       57272
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
Name: teacher_prefix, dtype: int64
```

In [13]:
```
project_data['teacher_prefix']=project_data['teacher_prefix'].str.replace('.','')
project_data['teacher_prefix']=project_data['teacher_prefix'].str.lower()
project_data['teacher_prefix'].value_counts()
```

Out[13]:
```
mrs        57272
ms         38955
mr         10648
teacher     2360
dr            13
Name: teacher_prefix, dtype: int64
```

project_grade_category

In [14]:
```
project_data['project_grade_category'].value_counts()
```

Out[14]:
```
Grades PreK-2    44225
Grades 3-5       37137
Grades 6-8       16923
Grades 9-12      10963
Name: project_grade_category, dtype: int64
```

In [15]:
```
print(project_data['project_grade_category'].isnull().values.any())
print("Number of nan values", project_data['project_grade_category'].isnull().values.sum())
```

```
False
Number of nan values 0
```

In [16]:
```
project_data['project_grade_category']=project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category']=project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category']=project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

Out[16]:
```
grades_prek_2    44225
grades_3_5       37137
grades_6_8       16923
grades_9_12      10963
Name: project_grade_category, dtype: int64
```

school_state

In [17]:
```
project_data['school_state'].value_counts()
```

Out[17]:
CA	15388
TX	7396
NY	7318
FL	6185
NC	5091
IL	4350
GA	3963
SC	3936
MI	3161
PA	3109
IN	2620
MO	2576
OH	2467
LA	2394
MA	2389
WA	2334
OK	2276
NJ	2237
AZ	2147
VA	2045
WI	1827
AL	1762
UT	1731
TN	1688
CT	1663
MD	1514
NV	1367
MS	1323
KY	1304
OR	1242
MN	1208
CO	1111
AR	1049
ID	693
IA	666
KS	634
NM	557
DC	516
HI	507
ME	505
WV	503

```
NH      348
AK      345
DE      343
NE      309
SD      300
RI      285
MT      245
ND      143
WY       98
VT       80
Name: school_state, dtype: int64
```

In [18]:
```
project_data['school_state'].isnull().values.any()
```

Out[18]:
```
False
```

In [19]:
```
project_data['school_state']=project_data['school_state'].str.lower()
project_data['school_state'].value_counts()
```

```
Out[19]: ca    15388
         tx     7396
         ny     7318
         fl     6185
         nc     5091
         il     4350
         ga     3963
         sc     3936
         mi     3161
         pa     3109
         in     2620
         mo     2576
         oh     2467
         la     2394
         ma     2389
         wa     2334
         ok     2276
         nj     2237
         az     2147
         va     2045
         wi     1827
         al     1762
         ut     1731
         tn     1688
         ct     1663
         md     1514
         nv     1367
         ms     1323
         ky     1304
         or     1242
         mn     1208
         co     1111
         ar     1049
         id      693
         ia      666
         ks      634
         nm      557
         dc      516
         hi      507
         me      505
         wv      503
```

```
nh      348
ak      345
de      343
ne      309
sd      300
ri      285
mt      245
nd      143
wy       98
vt       80
Name: school_state, dtype: int64
```

project_subject_categories

In [20]:
```
project_data['project_subject_categories'].value_counts()
```

Out[20]:

Literacy & Language	23655
Math & Science	17072
Literacy & Language, Math & Science	14636
Health & Sports	10177
Music & The Arts	5180
Special Needs	4226
Literacy & Language, Special Needs	3961
Applied Learning	3771
Math & Science, Literacy & Language	2289
Applied Learning, Literacy & Language	2191
History & Civics	1851
Math & Science, Special Needs	1840
Literacy & Language, Music & The Arts	1757
Math & Science, Music & The Arts	1642
Applied Learning, Special Needs	1467
History & Civics, Literacy & Language	1421
Health & Sports, Special Needs	1391
Warmth, Care & Hunger	1309
Math & Science, Applied Learning	1220
Applied Learning, Math & Science	1052
Literacy & Language, History & Civics	809
Health & Sports, Literacy & Language	803
Applied Learning, Music & The Arts	758
Math & Science, History & Civics	652
Literacy & Language, Applied Learning	636
Applied Learning, Health & Sports	608
Math & Science, Health & Sports	414
History & Civics, Math & Science	322
History & Civics, Music & The Arts	312
Special Needs, Music & The Arts	302
Health & Sports, Math & Science	271
History & Civics, Special Needs	252
Health & Sports, Applied Learning	192
Applied Learning, History & Civics	178
Health & Sports, Music & The Arts	155
Music & The Arts, Special Needs	138
Literacy & Language, Health & Sports	72
Health & Sports, History & Civics	43
History & Civics, Applied Learning	42
Special Needs, Health & Sports	42
Special Needs, Warmth, Care & Hunger	23

```
Health & Sports, Warmth, Care & Hunger          23
Music & The Arts, Health & Sports               19
Music & The Arts, History & Civics              18
History & Civics, Health & Sports               13
Math & Science, Warmth, Care & Hunger           11
Applied Learning, Warmth, Care & Hunger         10
Music & The Arts, Applied Learning              10
Literacy & Language, Warmth, Care & Hunger       9
Music & The Arts, Warmth, Care & Hunger          2
History & Civics, Warmth, Care & Hunger          1
Name: project_subject_categories, dtype: int64
```

In [21]:
```
print(project_data['project_subject_categories'].isnull().values.any())
print("Number of nan values", project_data['project_subject_categories'].isnull().values.sum())
```

```
False
Number of nan values 0
```

In [22]:
```
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.replace(' The ','')
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.replace(' ','')
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.replace('&','_')
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.replace(',','_')
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.lower()
project_data['project_subject_categories'].value_counts()
```

Out[22]:

literacy_language	23655
math_science	17072
literacy_language_math_science	14636
health_sports	10177
music_arts	5180
specialneeds	4226
literacy_language_specialneeds	3961
appliedlearning	3771
math_science_literacy_language	2289
appliedlearning_literacy_language	2191
history_civics	1851
math_science_specialneeds	1840
literacy_language_music_arts	1757
math_science_music_arts	1642
appliedlearning_specialneeds	1467
history_civics_literacy_language	1421
health_sports_specialneeds	1391
warmth_care_hunger	1309
math_science_appliedlearning	1220
appliedlearning_math_science	1052
literacy_language_history_civics	809
health_sports_literacy_language	803
appliedlearning_music_arts	758
math_science_history_civics	652
literacy_language_appliedlearning	636
appliedlearning_health_sports	608
math_science_health_sports	414
history_civics_math_science	322
history_civics_music_arts	312
specialneeds_music_arts	302
health_sports_math_science	271
history_civics_specialneeds	252
health_sports_appliedlearning	192
appliedlearning_history_civics	178
health_sports_music_arts	155
music_arts_specialneeds	138
literacy_language_health_sports	72
health_sports_history_civics	43
history_civics_appliedlearning	42
specialneeds_health_sports	42
specialneeds_warmth_care_hunger	23

```
health_sports_warmth_care_hunger          23
music_arts_health_sports                  19
music_arts_history_civics                 18
history_civics_health_sports              13
math_science_warmth_care_hunger           11
music_arts_appliedlearning                10
appliedlearning_warmth_care_hunger        10
literacy_language_warmth_care_hunger       9
music_arts_warmth_care_hunger              2
history_civics_warmth_care_hunger          1
Name: project_subject_categories, dtype: int64
```

project_subject_subcategories

In [23]:
```
project_data['project_subject_subcategories'].value_counts()
```

Out[23]:
```
Literacy                                9486
Literacy, Mathematics                   8325
Literature & Writing, Mathematics       5923
Literacy, Literature & Writing          5571
Mathematics                             5379
                                        ...
ESL, Economics                             1
Gym & Fitness, Warmth, Care & Hunger       1
Other, Warmth, Care & Hunger               1
Economics, Foreign Languages               1
Gym & Fitness, Social Sciences             1
Name: project_subject_subcategories, Length: 401, dtype: int64
```

In [24]:
```
print(project_data['project_subject_subcategories'].isnull().values.any())
print("Number of nan values", project_data['project_subject_subcategories'].isnull().values.sum())
```

```
False
Number of nan values 0
```

In [25]:
```
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace(' The ','')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace(' ','')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace('&','_')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.replace(',','_')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.lower()
project_data['project_subject_subcategories'].value_counts()
```

Out[25]:
```
literacy                                9486
literacy_mathematics                    8325
literature_writing_mathematics          5923
literacy_literature_writing             5571
mathematics                             5379
                                        ...
parentinvolvement_warmth_care_hunger       1
college_careerprep_warmth_care_hunger      1
other_warmth_care_hunger                   1
communityservice_financialliteracy         1
economics_nutritioneducation               1
Name: project_subject_subcategories, Length: 401, dtype: int64
```

1.3 Preporcessing Text Data

project_essay

In [26]:
```
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [27]:
```
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'furthe
r',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'mor
e',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're',
 \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "were
n't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [28]:
```
from tqdm import tqdm
def preprocess_text(text_data):
    preprocessed_text = []
    # tqdm is for printing the status bar
    for sentance in tqdm(text_data):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('\\"', ' ')
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        # https://gist.github.com/sebleier/554280
        sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
        preprocessed_text.append(sent.lower().strip())
    return preprocessed_text
```

In [29]:
```
project_data["essay"]=project_data["project_essay_1"].map(str) +\
                      project_data["project_essay_2"].map(str) +\
                      project_data["project_essay_3"].map(str) +\
                      project_data["project_essay_4"].map(str)
```

In [30]:
```
print("Printing some random essays")
print(9,project_data["essay"].values[9])
print('-'*100)
print(34,project_data["essay"].values[34])
print('-'*100)
print(147,project_data["essay"].values[147])
```

Printing some random essays
9 Over 95% of my students are on free or reduced lunch. I have a few who are homeless, but despite that, they come t
o school with an eagerness to learn. My students are inquisitive eager learners who embrace the challenge of not ha
ving great books and other resources every day. Many of them are not afforded the opportunity to engage with these
big colorful pages of a book on a regular basis at home and they don't travel to the public library. \r\nIt is my du
ty as a teacher to do all I can to provide each student an opportunity to succeed in every aspect of life. \r\nReadin
g is Fundamental! My students will read these books over and over again while boosting their comprehension skills. Th
ese books will be used for read alouds, partner reading and for Independent reading. \r\nThey will engage in reading
to build their \"Love for Reading\" by reading for pure enjoyment. They will be introduced to some new authors as wel
l as some old favorites. I want my students to be ready for the 21st Century and know the pleasure of holding a good
hard back book in hand. There's nothing like a good book to read! \r\nMy students will soar in Reading, and more bec
ause of your consideration and generous funding contribution. This will help build stamina and prepare for 3rd grade.
Thank you so much for reading our proposal!nannan
---
34 My students mainly come from extremely low-income families, and the majority of them come from homes where both pa
rents work full time. Most of my students are at school from 7:30 am to 6:00 pm (2:30 to 6:00 pm in the after-school
program), and they all receive free and reduced meals for breakfast and lunch. \r\n\r\n\r\nI want my students to feel
as comfortable in my classroom as they do at home. Many of my students take on multiple roles both at home as well as
in school. They are sometimes the caretakers of younger siblings, cooks, babysitters, academics, friends, and most of
all, they are developing who they are going to become as adults. I consider it an essential part of my job to model
helping others gain knowledge in a positive manner. As a result, I have a community of students who love helping each
other in and outside of the classroom. They consistently look for opportunities to support each other's learning in a
kind and helpful way.I am excited to be experimenting with alternative seating in my classroom this school year. Stud
ies have shown that giving students the option of where they sit in a classroom increases focus as well as motivatio
n. \r\n\r\nBy allowing students choice in the classroom, they are able to explore and create in a welcoming environm
ent. Alternative classroom seating has been experimented with more frequently in recent years. I believe (along with
many others), that every child learns differently. This does not only apply to how multiplication is memorized, or a
paper is written, but applies to the space in which they are asked to work. I have had students in the past ask \"Can
I work in the library? Can I work on the carpet?\" My answer was always, \"As long as you're learning, you can work w
herever you want!\" \r\n\r\nWith the yoga balls and the lap-desks, I will be able to increase the options for seating
in my classroom and expand its imaginable space.nannan
---
147 My students are eager to learn and make their mark on the world.\r\n\r\nThey come from a Title 1 school and need
extra love.\r\n\r\nMy fourth grade students are in a high poverty area and still come to school every day to get thei
r education. I am trying to make it fun and educational for them so they can get the most out of their schooling. I c
reated a caring environment for the students to bloom! They deserve the best.\r\nThank you!\r\nI am requesting 1 Chro
mebook to access online interventions, differentiate instruction, and get extra practice. The Chromebook will be used
to supplement ELA and math instruction. Students will play ELA and math games that are engaging and fun, as well as p
articipate in assignments online. This in turn will help my students improve their skills. Having a Chromebook in the
classroom would not only allow students to use the programs at their own pace, but would ensure more students are get
ting adequate time to use the programs. The online programs have been especially beneficial to my students with speci

```
al needs. They are able to work at their level as well as be challenged with some different materials. This is making
these students more confident in their abilities.\r\n\r\nThe Chromebook would allow my students to have daily access
to computers and increase their computing skills.\r\nThis will change their lives for the better as they become more
successful in school. Having access to technology in the classroom would help bridge the achievement gap.nannan
```

In [31]:
```
preprocessed_essays = preprocess_text(project_data['essay'].values)
```

```
100%|██████████| 109248/109248 [01:21<00:00, 1340.61it/s]
```

In [32]:
```
print("printing some random essay")
print(9, preprocessed_essays[9])
print('-'*50)
print(34, preprocessed_essays[34])
print('-'*50)
print(147, preprocessed_essays[147])
```

```
printing some random essay
9 95 students free reduced lunch homeless despite come school eagerness learn students inquisitive eager learners emb
race challenge not great books resources every day many not afforded opportunity engage big colorful pages book regul
ar basis home not travel public library duty teacher provide student opportunity succeed every aspect life reading fu
ndamental students read books boosting comprehension skills books used read alouds partner reading independent readin
g engage reading build love reading reading pure enjoyment introduced new authors well old favorites want students re
ady 21st century know pleasure holding good hard back book hand nothing like good book read students soar reading con
sideration generous funding contribution help build stamina prepare 3rd grade thank much reading proposal nannan
--------------------------------------------------
34 students mainly come extremely low income families majority come homes parents work full time students school 7 30
6 00 pm 2 30 6 00 pm school program receive free reduced meals breakfast lunch want students feel comfortable classro
om home many students take multiple roles home well school sometimes caretakers younger siblings cooks babysitters ac
ademics friends developing going become adults consider essential part job model helping others gain knowledge positi
ve manner result community students love helping outside classroom consistently look opportunities support learning k
ind helpful way excited experimenting alternative seating classroom school year studies shown giving students option
sit classroom increases focus well motivation allowing students choice classroom able explore create welcoming enviro
nment alternative classroom seating experimented frequently recent years believe along many others every child learns
differently not apply multiplication memorized paper written applies space asked work students past ask work library
work carpet answer always long learning work wherever want yoga balls lap desks able increase options seating classro
om expand imaginable space nannan
--------------------------------------------------
147 students eager learn make mark world come title 1 school need extra love fourth grade students high poverty area
still come school every day get education trying make fun educational get schooling created caring environment studen
ts bloom deserve best thank requesting 1 chromebook access online interventions differentiate instruction get extra p
ractice chromebook used supplement ela math instruction students play ela math games engaging fun well participate as
signments online turn help students improve skills chromebook classroom would not allow students use programs pace wo
uld ensure students getting adequate time use programs online programs especially beneficial students special needs a
ble work level well challenged different materials making students confident abilities chromebook would allow student
s daily access computers increase computing skills change lives better become successful school access technology cla
ssroom would help bridge achievement gap nannan
```

In [33]:
```
#adding processed essays to project_data
project_data['processed_essay']=preprocessed_essays
```

project_title

In [34]:
```
project_data['project_title'].head(5)
```

Out[34]:
```
0     Educational Support for English Learners at Home
1                Wanted: Projector for Hungry Learners
2    Soccer Equipment for AWESOME Middle School Stu...
3                               Techie Kindergarteners
4                               Interactive Math Tools
Name: project_title, dtype: object
```

In [35]:
```
print("printing some random titles")
print(9, project_data['project_title'].values[9])
print(34, project_data['project_title'].values[34])
print(147, project_data['project_title'].values[147])
```

```
printing some random titles
9 Just For the Love of Reading--\r\nPure Pleasure
34 \"Have A Ball!!!\"
147 Who needs a Chromebook?\r\nWE DO!!
```

In [36]:
```
from tqdm import tqdm
def preprocess_text(text_data):
    preprocessed_text = []
    # tqdm is for printing the status bar
    for sentance in tqdm(text_data):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('\\"', ' ')
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        # https://gist.github.com/sebleier/554280
        sent = ' '.join(e for e in sent.split() )
        preprocessed_text.append(sent.lower().strip())
    return preprocessed_text
```

In [37]:
```
preprocessed_titles = preprocess_text(project_data['project_title'].values)
```

```
100%|██████████| 109248/109248 [00:01<00:00, 58993.96it/s]
```

In [38]:
```
print("printing some random titles")
print(9, preprocessed_titles[9])
print(34, preprocessed_titles[34])
print(147, preprocessed_titles[147])
```

```
printing some random titles
9 just for the love of reading pure pleasure
34 have a ball
147 who needs a chromebook we do
```

In [39]:
```
#adding processed essays to project_data
project_data['preprocessed_titles']=preprocessed_titles
```

1.4 Preprocessing Numerical Features

In [40]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[40]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [41]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [42]:
```
project_data['price'].head()
```

Out[42]:
```
0    154.60
1    299.00
2    516.85
3    232.90
4     67.98
Name: price, dtype: float64
```

In [43]:
```
project_data.columns.values
```

Out[43]:
```
array(['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', 'essay', 'processed_essay',
       'preprocessed_titles', 'price', 'quantity'], dtype=object)
```

Removing unnecessary columns

In [44]:
```
project_data = project_data.drop(project_data.columns[[0,1,2,5,9,10,11,12,13,14,17,21]], axis=1)
```

In [45]:
```
project_data.head()
```

Out[45]:

	teacher_prefix	school_state	project_grade_category	project_subject_categories	project_subject_subcategories	teacher_numb
0	mrs	in	grades_prek_2	literacy_language	esl_literacy	0
1	mr	fl	grades_6_8	history_civics_health_sports	civics_government_teamsports	7
2	ms	az	grades_6_8	health_sports	health_wellness_teamsports	1
3	mrs	ky	grades_prek_2	literacy_language_math_science	literacy_mathematics	4
4	mrs	tx	grades_prek_2	math_science	mathematics	1

In [46]:
```
#renaming some columns
project_data = project_data.rename(index=str, columns=
                                   {'project_subject_categories':'clean_categories',
                                    'project_subject_subcategories':'clean_subcategories'})
```

In [47]:
```
#changing position of columns : https://stackoverflow.com/questions/41968732/set-order-of-columns-in-pandas-dataframe
project_data = project_data[['processed_essay', 'preprocessed_titles','teacher_prefix','project_grade_category',
                             'school_state','clean_categories','clean_subcategories',
                             'teacher_number_of_previously_posted_projects','price','project_is_approved']]
```

converting dataframe to csv

In [48]:
```
project_data.to_csv("preprocessed_data.csv",index=False)
```

2. GBDT

2.1 Loading Data

In [49]:
```
data=pd.read_csv("preprocessed_data.csv",nrows=50000)
```

In [50]:
```
data.head()
```

Out[50]:

	processed_essay	preprocessed_titles	teacher_prefix	project_grade_category	school_state	clean_categories	
0	students english learners working english seco...	educational support for english learners at home	mrs	grades_prek_2	in	literacy_language	esl_lite
1	students arrive school eager learn polite gene...	wanted projector for hungry learners	mr	grades_6_8	fl	history_civics_health_sports	civics_
2	true champions not always ones win guts mia ha...	soccer equipment for awesome middle school stu...	ms	grades_6_8	az	health_sports	health_
3	work unique school filled esl english second l...	techie kindergarteners	mrs	grades_prek_2	ky	literacy_language_math_science	literacy
4	second grade classroom next year made around 2...	interactive math tools	mrs	grades_prek_2	tx	math_science	mathe

In [51]:
```
data.shape
```

Out[51]: (50000, 10)

2.2 Splitting data into Train and Test

In [52]:
```
y = data['project_is_approved'].values
X = data.drop(['project_is_approved'], axis=1)
```

In [53]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, stratify=y)
```

In [54]:
```
X_train.shape
```

Out[54]: (35000, 9)

In [55]:
```
X_test.shape
```

Out[55]: (15000, 9)

2.3 Vectorizing Text Data

Essays

TFIDF

In [ ]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

print("="*100)


vectorizer1 = TfidfVectorizer(min_df=5, max_features=5000)
vectorizer1.fit(X_train['processed_essay'].values) # fit has to happen only on train data


X_train_essay_tfidf = vectorizer1.transform(X_train['processed_essay'].values)
X_test_essay_tfidf = vectorizer1.transform(X_test['processed_essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
print("="*100)
```

```
(35000, 9) (35000,)
(15000, 9) (15000,)
====================================================================================================
After vectorizations
(35000, 5000) (35000,)
(15000, 5000) (15000,)
====================================================================================================
```

TFIDF W2V

In [56]:
```
with open("/content/drive/MyDrive/GBDT/GBDT Assignment/glove_vectors", 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [57]:
```
tfidf_model = TfidfVectorizer(min_df=5, max_features=5000)
tfidf_model.fit(X_train['processed_essay'].values)
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

TFIDF W2V X_train

In [58]:
```
train_tfidf_w2v_vectors = [];
for sentence in tqdm(X_train['processed_essay']):
    vector = np.zeros(300)
    tf_idf_weight =0;
    for word in sentence.split():
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_vectors.append(vector)

print(len(train_tfidf_w2v_vectors))
print(len(train_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 35000/35000 [01:03<00:00, 547.75it/s]

35000
300
```

TFIDF W2V X_test

In [59]:
```
test_tfidf_w2v_vectors = [];
for sentence in tqdm(X_test['processed_essay']):
    vector = np.zeros(300)
    tf_idf_weight =0;
    for word in sentence.split():
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_vectors.append(vector)

print(len(test_tfidf_w2v_vectors))
print(len(test_tfidf_w2v_vectors[0]))
```

```
100%|██████████| 15000/15000 [00:27<00:00, 550.39it/s]

15000
300
```

project title

TFIDF

In [ ]:
```
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)

print("="*100)


vectorizer10 = TfidfVectorizer(min_df=5)
vectorizer10.fit(X_train['preprocessed_titles'].values)  # fit has to happen only on train data


X_train_title_tfidf = vectorizer10.transform(X_train['preprocessed_titles'].values)
X_test_title_tfidf = vectorizer10.transform(X_test['preprocessed_titles'].values)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_test_title_tfidf.shape, y_test.shape)
print("="*100)
```

```
(35000, 9) (35000,)
(15000, 9) (15000,)
====================================================================================================
After vectorizations
(35000, 2702) (35000,)
(15000, 2702) (15000,)
====================================================================================================
```

TFIDF W2V

In [60]:
```
with open("/content/drive/MyDrive/GBDT/GBDT Assignment/glove_vectors", 'rb') as f:

    glove_words =  set(model.keys())
```

In [61]:
```
tfidf_model1 = TfidfVectorizer(min_df=5)
tfidf_model1.fit(X_train['preprocessed_titles'].values.astype('U'))
dictionary = dict(zip(tfidf_model1.get_feature_names(), list(tfidf_model1.idf_)))
tfidf_words = set(tfidf_model1.get_feature_names())
```

TFIDF W2V X_train

In [62]:
```
train_title_w2v_vectors = [];
for sentence in tqdm(X_train['preprocessed_titles']):
    vector = np.zeros(300)
    tf_idf_weight =0;
    for word in sentence.split():
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_title_w2v_vectors.append(vector)

print(len(train_title_w2v_vectors))
print(len(train_title_w2v_vectors[0]))
```

```
100%|██████████| 35000/35000 [00:01<00:00, 24168.63it/s]

35000
300
```

TFIDF W2V X_test

In [63]:
```
test_title_w2v_vectors = [];
for sentence in tqdm(X_test['preprocessed_titles']):
    vector = np.zeros(300)
    tf_idf_weight =0;
    for word in sentence.split():
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_title_w2v_vectors.append(vector)

print(len(test_title_w2v_vectors))
print(len(test_title_w2v_vectors[0]))
```

```
100%|██████████| 15000/15000 [00:00<00:00, 19107.37it/s]

15000
300
```

2.4 Encoding Numerical Feature

Price

In [64]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()


X_train_price_norm = normalizer.fit_transform(X_train['price'].values.reshape(1,-1))
X_test_price_norm = normalizer.fit_transform(X_test['price'].values.reshape(1,-1))



X_train_price_norm=X_train_price_norm.reshape(-1,1)
X_test_price_norm=X_test_price_norm.reshape(-1,1)

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(35000, 1) (35000,)
(15000, 1) (15000,)
====================================================================================================
```

teacher_number_of_previously_posted_projects

In [65]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()

X_train_teachernumber_norm = normalizer.fit_transform(X_train['teacher_number_of_previously_posted_projects'].values.r
eshape(1,-1))
X_test_teachernumber_norm = normalizer.fit_transform(X_test['teacher_number_of_previously_posted_projects'].values.res
hape(1,-1))



X_train_teachernumber_norm=X_train_price_norm.reshape(-1,1)
X_test_teachernumber_norm=X_test_price_norm.reshape(-1,1)

print("After vectorizations")
print(X_train_teachernumber_norm.shape, y_train.shape)
print(X_test_teachernumber_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(35000, 1) (35000,)
(15000, 1) (15000,)
====================================================================================================
```

2.5 Sentiment Scores

In [ ]:
```
import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

Out[ ]:
```
True
```

In [ ]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

In [ ]:
```
sid = SentimentIntensityAnalyzer()
essay=X_train['processed_essay']
essay_sentiment1=[]
essay_sentiment2=[]
essay_sentiment3=[]
essay_sentiment4=[]
for i in tqdm(essay):
    score = sid.polarity_scores(i)
    essay_sentiment1.append(score['neg'])
    essay_sentiment2.append(score['neu'])
    essay_sentiment3.append(score['pos'])
    essay_sentiment4.append(score['compound'])
X_train['neg_sentiment_train'] = essay_sentiment1
X_train['neu_sentiment_train'] = essay_sentiment2
X_train['pos_sentiment_train'] = essay_sentiment3
X_train['compound_sentiment_train'] = essay_sentiment4
```

```
100%|██████████| 35000/35000 [01:10<00:00, 495.26it/s]
```

In [ ]:
```
neg_sentiment_train=X_train['neg_sentiment_train'].values.reshape(-1,1)
neu_sentiment_train=X_train['neu_sentiment_train'].values.reshape(-1,1)
pos_sentiment_train=X_train['pos_sentiment_train'].values.reshape(-1,1)
compound_sentiment_train=X_train['compound_sentiment_train'].values.reshape(-1,1)
```

In [ ]:
```
sid = SentimentIntensityAnalyzer()
essay=X_test['processed_essay']
essay_sentiment1=[]
essay_sentiment2=[]
essay_sentiment3=[]
essay_sentiment4=[]
for i in tqdm(essay):
    score = sid.polarity_scores(i)
    essay_sentiment1.append(score['neg'])
    essay_sentiment2.append(score['neu'])
    essay_sentiment3.append(score['pos'])
    essay_sentiment4.append(score['compound'])
X_test['neg_sentiment_test'] = essay_sentiment1
X_test['neu_sentiment_test'] = essay_sentiment2
X_test['pos_sentiment_test'] = essay_sentiment3
X_test['compound_sentiment_test'] = essay_sentiment4
```

```
100%|██████████| 15000/15000 [00:29<00:00, 503.39it/s]
```

In [ ]:
```
neg_sentiment_test=X_test['neg_sentiment_test'].values.reshape(-1,1)
neu_sentiment_test=X_test['neu_sentiment_test'].values.reshape(-1,1)
pos_sentiment_test=X_test['pos_sentiment_test'].values.reshape(-1,1)
compound_sentiment_test=X_test['compound_sentiment_test'].values.reshape(-1,1)
```

2.6 Encoding Categorical Features

Response Coding

In [66]:
```
project=y_train.tolist()
```

In [67]:
```
new_xytrain=X_train[["teacher_prefix","project_grade_category","school_state","clean_categories","clean_subcategories"
]].copy()
```

In [68]:
```
new_xytrain['project_is_approved']=project
```

In [69]:
```
new_xytrain.shape
```

Out[69]: (35000, 6)

In [70]:
```
new_xytrain.head()
```

Out[70]:

	teacher_prefix	project_grade_category	school_state	clean_categories	clean_subcategories	project_is_approve
9863	mrs	grades_9_12	tx	music_arts	visualarts	1
49765	mrs	grades_3_5	pa	literacy_language_math_science	literacy_mathematics	1
9175	ms	grades_prek_2	il	literacy_language	literacy	0
28982	mrs	grades_6_8	al	literacy_language	literacy	0
32638	mrs	grades_prek_2	ms	warmth_care_hunger	warmth_care_hunger	1

In [71]:
```
new_xytest=X_test[["teacher_prefix","project_grade_category","school_state","clean_categories","clean_subcategories"]]
.copy()
```

In [72]:
```
projecttest=y_test.tolist()
```

In [73]:
```
new_xytest['project_is_approved']=projecttest
```

In [74]:
```
new_xytest.shape
```

Out[74]: (15000, 6)

In [75]:
```
new_xytest.head()
```

Out[75]:

	teacher_prefix	project_grade_category	school_state	clean_categories	clean_subcategories	project_is_app
27962	mrs	grades_prek_2	ca	literacy_language_music_arts	esl_performingarts	1
33191	ms	grades_3_5	ny	specialneeds	specialneeds	1
47451	mrs	grades_prek_2	nj	literacy_language	literacy_literature_writing	1
19124	mrs	grades_prek_2	ar	literacy_language	literacy	1
15794	mrs	grades_6_8	la	math_science	appliedsciences_mathematics	0

In [76]:
```
def trainResponseEncoding(working_data, cat_type):

    class_0 = working_data[working_data['project_is_approved']==0].groupby(cat_type).size()
    class_1 = working_data[working_data['project_is_approved']==1].groupby(cat_type).size()
    dict_0=class_0.to_dict()
    dict_1=class_1.to_dict()
    return dict_0,dict_1
```

In [77]:
```
def mergdict(dict_0,dict_1):
    d3 =dict_0.copy()
    d3.update(dict_1)
    for i, j in dict_0.items():
        for x, y in dict_1.items():
            if i == x:
                d3[i]=(j+y)
    return d3
```

In [78]:
```
def getResponseEncondingdata(data_0, data_1,d3, working_data, cat_type):
    cat=working_data[cat_type].unique().tolist()
    pos=[]
    neg=[]

    for i in cat:

        for p in d3.items():
            if(i==p[0]):
                for q in data_0.items():
                    if(i==q[0]):
                        neg.append((q[1])/(d3.get(i)))
                        break


                else:
                    neg.append(0)
                break
        else:
            neg.append(0.5)



    pos[:]=[1-x for x in neg]

    encoded_Pos_val = dict(zip(cat, pos))
    encoded_Neg_val = dict(zip(cat, neg))

    return encoded_Pos_val, encoded_Neg_val
```

In [79]:
```
#fit categorical train data
def fit():
    prefix_0,prefix_1=trainResponseEncoding(new_xytrain, 'teacher_prefix')
    grade_0,grade_1=trainResponseEncoding(new_xytrain, 'project_grade_category')
    state_0,state_1=trainResponseEncoding(new_xytrain, 'school_state')
    category_0,category_1=trainResponseEncoding(new_xytrain, 'clean_categories')
    subcat_0,subcat_1=trainResponseEncoding(new_xytrain, 'clean_subcategories')

    return prefix_0,prefix_1,grade_0,grade_1,state_0,state_1,category_0,category_1,subcat_0,subcat_1


prefix_0,prefix_1,grade_0,grade_1,state_0,state_1,category_0,category_1,subcat_0,subcat_1=fit()
```

In [80]:
```
#Create dictionary by merging features
def merging():
    pre3=mergdict(prefix_0,prefix_1)
    gra3=mergdict(grade_0,grade_1)
    sta3=mergdict(state_0,state_1)
    cat3=mergdict(category_0,category_1)
    sub3=mergdict(subcat_0,subcat_1)

    return pre3,gra3,sta3,cat3,sub3

pre3,gra3,sta3,cat3,sub3=merging()
```

In [81]:
```
#transform train data
def transformtrain():
    #trainset
    train_pre_pos,train_pre_neg=getResponseEncondingdata(prefix_0,prefix_1,pre3, new_xytrain, 'teacher_prefix')
    train_grade_pos,train_grade_neg=getResponseEncondingdata(grade_0,grade_1,gra3, new_xytrain, 'project_grade_categor
y')
    train_state_pos,train_state_neg=getResponseEncondingdata(state_0,state_1,sta3, new_xytrain, 'school_state')
    train_cat_pos,train_cat_neg=getResponseEncondingdata(category_0,category_1,cat3, new_xytrain, 'clean_categories')
    train_sub_pos,train_sub_neg=getResponseEncondingdata(subcat_0,subcat_1,sub3, new_xytrain, 'clean_subcategories')

    return train_pre_pos,train_pre_neg,train_grade_pos,train_grade_neg,train_state_pos,train_state_neg,train_cat_pos,t
rain_cat_neg,train_sub_pos,train_sub_neg
```

In [82]:
```
#calling function
train_pre_pos,train_pre_neg,train_grade_pos,train_grade_neg,train_state_pos,train_state_neg,train_cat_pos,train_cat_ne
g,train_sub_pos,train_sub_neg=transformtrain()
```

In [83]:
```
#transform test data
def transformtest():
    test_pre_pos,test_pre_neg=getResponseEncondingdata(prefix_0,prefix_1,pre3, new_xytest, 'teacher_prefix')
    test_grade_pos,test_grade_neg=getResponseEncondingdata(grade_0,grade_1,gra3, new_xytest, 'project_grade_category')
    test_state_pos,test_state_neg=getResponseEncondingdata(state_0,state_1,sta3, new_xytest, 'school_state')
    test_cat_pos,test_cat_neg=getResponseEncondingdata(category_0,category_1,cat3, new_xytest, 'clean_categories')
    test_sub_pos,test_sub_neg=getResponseEncondingdata(subcat_0,subcat_1,sub3, new_xytest, 'clean_subcategories')

    return test_pre_pos,test_pre_neg,test_grade_pos,test_grade_neg,test_state_pos,test_state_neg,test_cat_pos,test_cat
_neg,test_sub_pos,test_sub_neg
```

In [84]:
```
#calling function
test_pre_pos,test_pre_neg,test_grade_pos,test_grade_neg,test_state_pos,test_state_neg,test_cat_pos,test_cat_neg,test_s
ub_pos,test_sub_neg=transformtest()
```

In [85]:
```
new_xytrain['tr_pre_p'] = new_xytrain['teacher_prefix'].map(train_pre_pos)
new_xytrain['tr_pre_n'] = new_xytrain['teacher_prefix'].map(train_pre_neg)

new_xytrain['tr_gra_p'] = new_xytrain['project_grade_category'].map(train_grade_pos)
new_xytrain['tr_gra_n'] = new_xytrain['project_grade_category'].map(train_grade_neg)

new_xytrain['tr_state_p'] = new_xytrain['school_state'].map(train_state_pos)
new_xytrain['tr_state_n'] = new_xytrain['school_state'].map(train_state_neg)

new_xytrain['tr_cat_p'] = new_xytrain['clean_categories'].map(train_cat_pos)
new_xytrain['tr_cat_n'] = new_xytrain['clean_categories'].map(train_cat_neg)

new_xytrain['tr_sub_p'] = new_xytrain['clean_subcategories'].map(train_sub_pos)
new_xytrain['tr_sub_n'] = new_xytrain['clean_subcategories'].map(train_sub_neg)
```

In [86]:
```
tr_pre_p=new_xytrain['tr_pre_p'].values.reshape(-1,1)
tr_pre_n=new_xytrain['tr_pre_n'].values.reshape(-1,1)

tr_gra_p=new_xytrain['tr_gra_p'].values.reshape(-1,1)
tr_gra_n=new_xytrain['tr_gra_n'].values.reshape(-1,1)

tr_state_p=new_xytrain['tr_state_p'].values.reshape(-1,1)
tr_state_n=new_xytrain['tr_state_n'].values.reshape(-1,1)

tr_cat_p=new_xytrain['tr_cat_p'].values.reshape(-1,1)
tr_cat_n=new_xytrain['tr_cat_n'].values.reshape(-1,1)

tr_sub_p=new_xytrain['tr_sub_p'].values.reshape(-1,1)
tr_sub_n=new_xytrain['tr_sub_n'].values.reshape(-1,1)
```

In [87]:
```
new_xytest['te_pre_p'] = new_xytest['teacher_prefix'].map(test_pre_pos)
new_xytest['te_pre_n'] = new_xytest['teacher_prefix'].map(test_pre_neg)

new_xytest['te_gra_p'] = new_xytest['project_grade_category'].map(test_grade_pos)
new_xytest['te_gra_n'] = new_xytest['project_grade_category'].map(test_grade_neg)

new_xytest['te_state_p'] = new_xytest['school_state'].map(test_state_pos)
new_xytest['te_state_n'] = new_xytest['school_state'].map(test_state_neg)

new_xytest['te_cat_p'] = new_xytest['clean_categories'].map(test_cat_pos)
new_xytest['te_cat_n'] = new_xytest['clean_categories'].map(test_cat_neg)

new_xytest['te_sub_p'] = new_xytest['clean_subcategories'].map(test_sub_pos)
new_xytest['te_sub_n'] = new_xytest['clean_subcategories'].map(test_sub_neg)
```

In [88]:
```
te_pre_p=new_xytest['te_pre_p'].values.reshape(-1,1)
te_pre_n=new_xytest['te_pre_n'].values.reshape(-1,1)

te_gra_p=new_xytest['te_gra_p'].values.reshape(-1,1)
te_gra_n=new_xytest['te_gra_n'].values.reshape(-1,1)

te_state_p=new_xytest['te_state_p'].values.reshape(-1,1)
te_state_n=new_xytest['te_state_n'].values.reshape(-1,1)

te_cat_p=new_xytest['te_cat_p'].values.reshape(-1,1)
te_cat_n=new_xytest['te_cat_n'].values.reshape(-1,1)

te_sub_p=new_xytest['te_sub_p'].values.reshape(-1,1)
te_sub_n=new_xytest['te_sub_n'].values.reshape(-1,1)
```

3. Merging all features

SET 1

In [ ]:
```
from scipy.sparse import hstack
X_tr=hstack((tr_pre_p,tr_pre_n,tr_gra_p,tr_gra_n,tr_state_p,tr_state_n,tr_cat_p,tr_cat_n,tr_sub_p,tr_sub_n,X_train_price_norm,X_train_teachernumber_norm,X_train_title_tfidf,X_train_essay_tfidf,neg_sentiment_train,neu_sentiment_train,pos_sentiment_train,compound_sentiment_train)).tocsr()
X_te=hstack((te_pre_p,te_pre_n,te_gra_p,te_gra_n,te_state_p,te_state_n,te_cat_p,te_cat_n,te_sub_p,te_sub_n,X_test_price_norm,X_test_teachernumber_norm,X_test_title_tfidf,X_test_essay_tfidf,neg_sentiment_test,neu_sentiment_test,pos_sentiment_test,compound_sentiment_test)).tocsr()
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(35000, 7718) (35000,)
(15000, 7718) (15000,)
====================================================================================================
```

SET 2

In [89]:
```
from scipy.sparse import hstack,csr_matrix
X_tr1=hstack((csr_matrix(tr_pre_p),tr_pre_n,tr_gra_p,tr_gra_n,tr_state_p,tr_state_n,tr_cat_p,tr_cat_n,tr_sub_p,tr_sub_n,X_train_price_norm,X_train_teachernumber_norm,train_title_w2v_vectors,train_tfidf_w2v_vectors)).tocsr()
X_te1=hstack((csr_matrix(te_pre_p),te_pre_n,te_gra_p,te_gra_n,te_state_p,te_state_n,te_cat_p,te_cat_n,te_sub_p,te_sub_n,X_test_price_norm,X_test_teachernumber_norm,test_title_w2v_vectors,test_tfidf_w2v_vectors)).tocsr()
print("Final Data matrix")
print(X_tr1.shape, y_train.shape)
print(X_te1.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(35000, 612) (35000,)
(15000, 612) (15000,)
====================================================================================================
```

Task - 1

1. **Apply GBDT on SET 1**

3.1.1 Hyper parameter Tuning

In [ ]:
```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import GradientBoostingClassifier

GBDT=GradientBoostingClassifier()

parameters={'max_depth':[1,5,10,50],'n_estimators':[5,10,100,200]}


clf =RandomizedSearchCV(GBDT, parameters, cv= 3, scoring='roc_auc',verbose=10,return_train_score=True,n_jobs=-1)
clf.fit(X_tr,y_train)

print('Best Score: ', clf.best_score_)
bestMaxDepth=clf.best_params_['max_depth']
bestEstimator=clf.best_params_['n_estimators']
print('bestMaxDepth:',clf.best_params_['max_depth'])
print('bestEstimator:',clf.best_params_['n_estimators'])
```

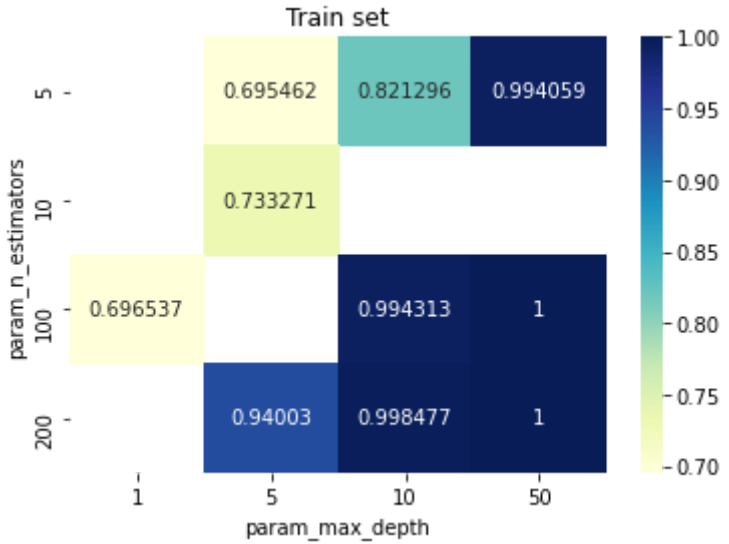
```
Fitting 3 folds for each of 10 candidates, totalling 30 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:   28.1s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:  1.4min
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:  7.6min
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed: 39.4min
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed: 57.6min
[Parallel(n_jobs=-1)]: Done  30 out of  30 | elapsed: 303.8min finished

Best Score:  0.6957839137386411
bestMaxDepth: 5
bestEstimator: 200
```

3.1.2 Representation of results

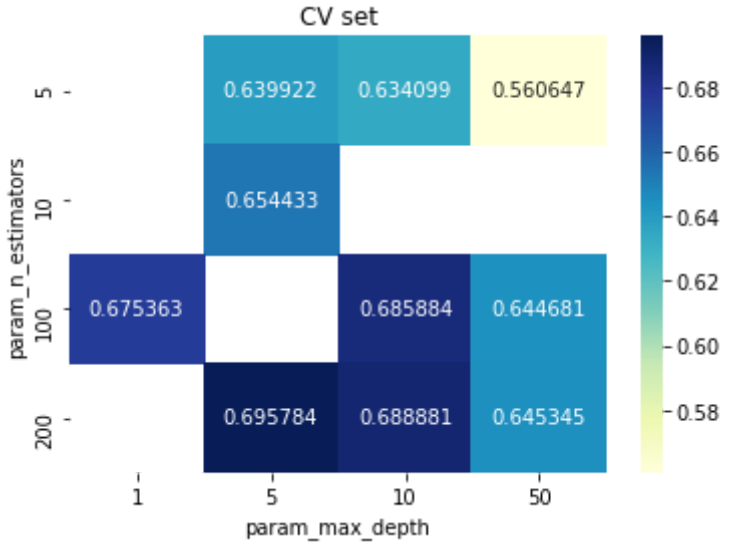
In [ ]:
```
trscore=clf.cv_results_['mean_train_score']
trscore.tolist()
testscore=clf.cv_results_['mean_test_score']
testscore.tolist()
sample=clf.cv_results_['param_n_estimators']
sample.tolist()
depth=clf.cv_results_['param_max_depth']
print(depth.tolist())
```

```
[5, 1, 50, 10, 10, 5, 10, 5, 50, 50]
```

In [ ]:
```
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe
df = pd.DataFrame({'trscore': trscore, 'param_n_estimators':sample, 'param_max_depth':depth})
result = df.pivot(index='param_n_estimators', columns='param_max_depth', values='trscore')
sns.heatmap(result, annot=True, fmt="g",cmap='YlGnBu')
plt.title("Train set")
plt.show()
```

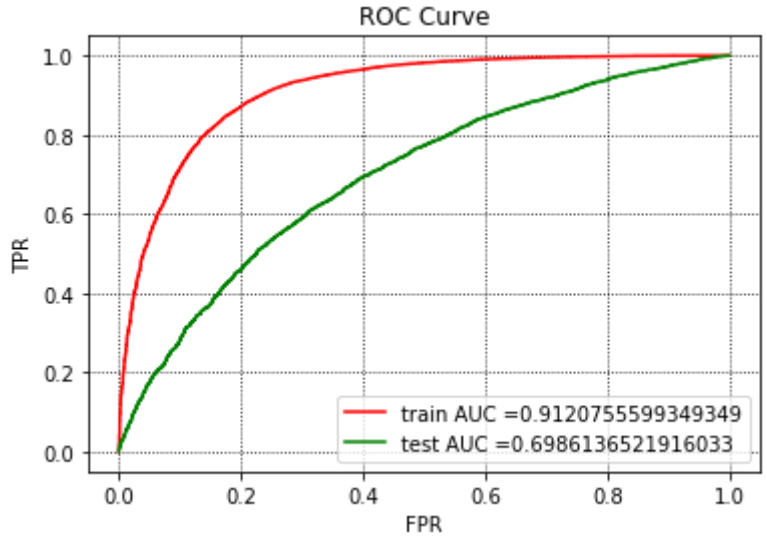


In [ ]:
```
#test result
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe
df = pd.DataFrame({'tescore': testscore, 'param_n_estimators':sample, 'param_max_depth':depth})
result = df.pivot(index='param_n_estimators', columns='param_max_depth', values='tescore')
sns.heatmap(result, annot=True, fmt="g",cmap="YlGnBu")
plt.title("CV set")
plt.show()
```



3.1.3 ROC CURVE

In [ ]:
```
gb_test = GradientBoostingClassifier(n_estimators=bestEstimator,max_depth=bestMaxDepth)
gb_test.fit(X_tr,y_train)

y_train_pred = gb_test.predict_proba( X_tr)[:, 1]
y_test_pred = gb_test.predict_proba(X_te)[:, 1]
train_fpr, train_tpr, tr_thresholds =  metrics.roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = metrics.roc_curve(y_test, y_test_pred)

#Plot curve :
ab=plt.subplot()
plt.plot(train_fpr, train_tpr,color='r',label="train AUC ="+str(metrics.auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr,color='g', label="test AUC ="+str(metrics.auc(test_fpr, test_tpr)))
plt.legend(loc='lower right')
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.grid(b=True, which='major', color='k', linestyle=':')
ab.set_facecolor("white")
plt.show()
```



3.1.4 Confusion Matrix

In [ ]:
```
#finding best threshold : https://stats.stackexchange.com/questions/123124/how-to-determine-the-optimal-threshold-for-a-classifier-and-generate-roc-curve
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_test_pred)
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print("Threshold value is:", np.round(optimal_threshold,3))
```

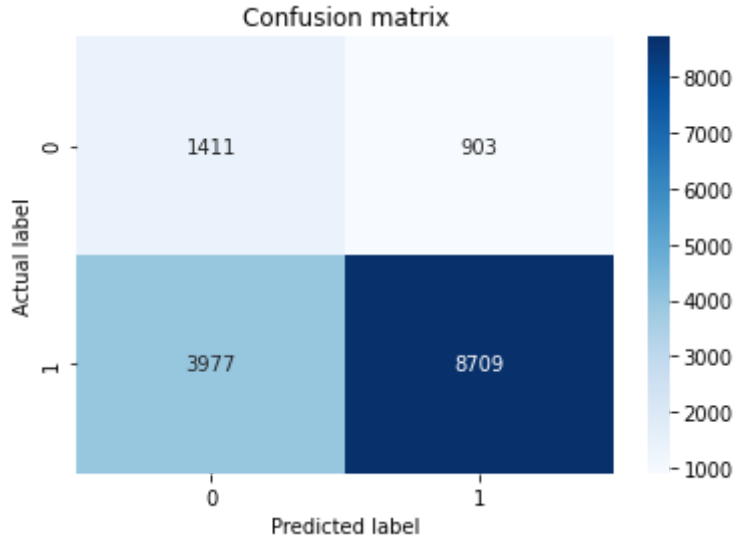
```
Threshold value is: 0.834
```

In [ ]:
```
def predict(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
prediction = predict(y_test_pred,optimal_threshold)
```

In [ ]:
```
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(y_test,prediction)
print('Confusion matrix : \n',matrix)
```

```
Confusion matrix :
 [[1411  903]
 [3977 8709]]
```

In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(matrix, annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted label')
plt.ylabel('Actual label')
plt.title('Confusion matrix')
plt.show()
```



1. **Apply GBDT on SET 2**

3.2.1 Hyper parameter Tuning

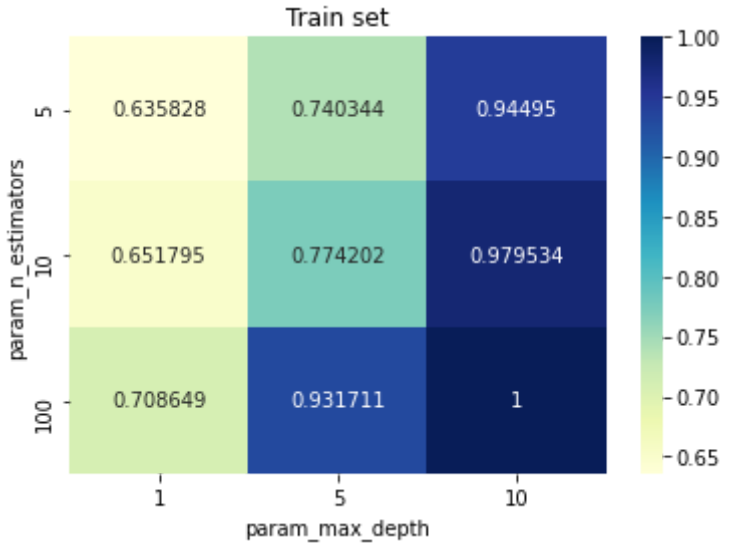
In [97]:
```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.ensemble import GradientBoostingClassifier

GBDT=GradientBoostingClassifier()

parameters={'max_depth':[1,5,10],'n_estimators':[5,10,100]}

clf1 =RandomizedSearchCV(GBDT, parameters, cv= 3, scoring='roc_auc',verbose=10,return_train_score=True,n_jobs=-1)
clf1.fit(X_tr1,y_train)


print('Best Score: ', clf1.best_score_)
bestMaxDepth=clf1.best_params_['max_depth']
bestEstimator=clf1.best_params_['n_estimators']
print('bestMaxDepth:',clf1.best_params_['max_depth'])
print('bestEstimator:',clf1.best_params_['n_estimators'])
```

```
Fitting 3 folds for each of 9 candidates, totalling 27 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done   1 tasks      | elapsed:   17.8s
[Parallel(n_jobs=-1)]: Done   4 tasks      | elapsed:   49.3s
[Parallel(n_jobs=-1)]: Done   9 tasks      | elapsed:  7.8min
[Parallel(n_jobs=-1)]: Done  14 tasks      | elapsed: 14.1min
[Parallel(n_jobs=-1)]: Done  21 tasks      | elapsed: 59.3min
[Parallel(n_jobs=-1)]: Done  27 out of  27 | elapsed: 183.7min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  27 out of  27 | elapsed: 183.7min finished
[Parallel(n_jobs=-1)]: Done  27 out of  27 | elapsed: 183.7min remaining:    0.0s
[Parallel(n_jobs=-1)]: Done  27 out of  27 | elapsed: 183.7min finished

Best Score:  0.6953575103744623
bestMaxDepth: 5
bestEstimator: 100
Best Score:  0.6953575103744623
bestMaxDepth: 5
bestEstimator: 100
```

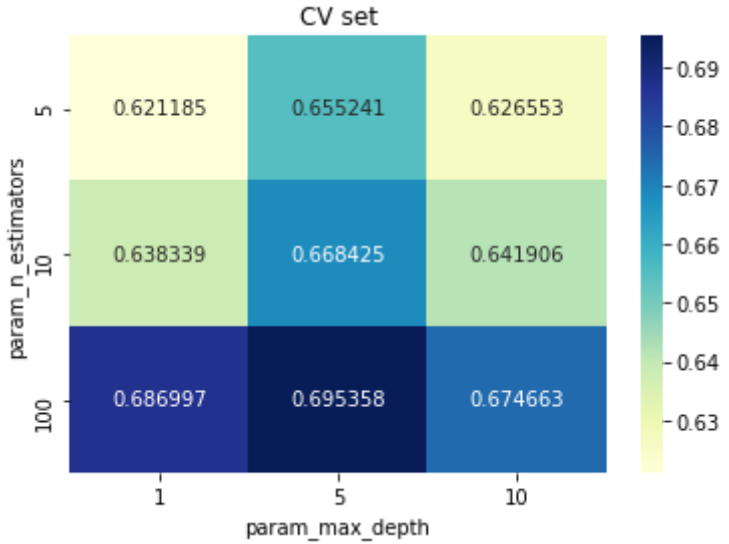
3.2.2 Representation of results

In [101]:
```
trscore=clf1.cv_results_['mean_train_score']
trscore.tolist()
testscore=clf1.cv_results_['mean_test_score']
testscore.tolist()
sample=clf1.cv_results_['param_n_estimators']
sample.tolist()
depth=clf1.cv_results_['param_max_depth']
```

In [104]:
```
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe
df = pd.DataFrame({'trscore': trscore, 'param_n_estimators':sample, 'param_max_depth':depth})
result = df.pivot(index='param_n_estimators', columns='param_max_depth', values='trscore')
sns.heatmap(result, annot=True, fmt="g",cmap='YlGnBu')
plt.title("Train set")
plt.show()
```

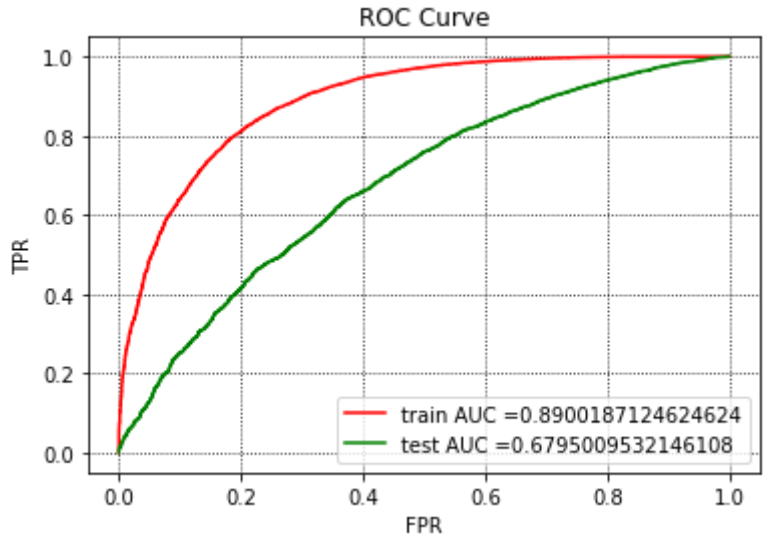


In [103]:
```
#test result
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe
df = pd.DataFrame({'tescore': testscore, 'param_n_estimators':sample, 'param_max_depth':depth})
result = df.pivot(index='param_n_estimators', columns='param_max_depth', values='tescore')
sns.heatmap(result, annot=True, fmt="g",cmap="YlGnBu")
plt.title("CV set")
plt.show()
```



3.2.3 ROC CURVE

In [105]:
```
gb_test = GradientBoostingClassifier(n_estimators=bestEstimator,max_depth=bestMaxDepth)
gb_test.fit(X_tr1,y_train)

y_train_pred = gb_test.predict_proba( X_tr1)[:, 1]
y_test_pred = gb_test.predict_proba(X_te1)[:, 1]
train_fpr, train_tpr, tr_thresholds =  metrics.roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = metrics.roc_curve(y_test, y_test_pred)

#Plot curve :
ab=plt.subplot()
plt.plot(train_fpr, train_tpr,color='r',label="train AUC ="+str(metrics.auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr,color='g', label="test AUC ="+str(metrics.auc(test_fpr, test_tpr)))
plt.legend(loc='lower right')
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC Curve")
plt.grid(b=True, which='major', color='k', linestyle=':')
ab.set_facecolor("white")
plt.show()
```



3.2.4 Confusion Matrix

In [106]:
```
#finding best threshold : https://stats.stackexchange.com/questions/123124/how-to-determine-the-optimal-threshold-for-a-classifier-and-generate-roc-curve
from sklearn.metrics import roc_curve, auc
fpr, tpr, thresholds = roc_curve(y_test, y_test_pred)
optimal_idx = np.argmax(tpr - fpr)
optimal_threshold = thresholds[optimal_idx]
print("Threshold value is:", np.round(optimal_threshold,3))
```

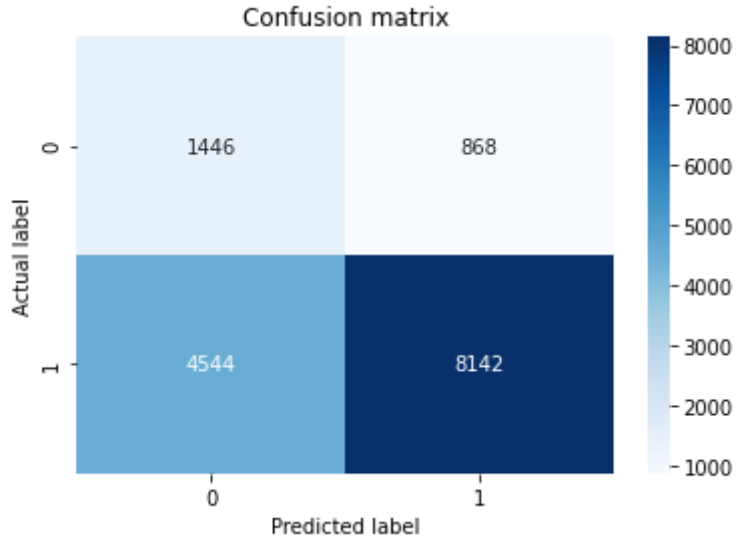
```
Threshold value is: 0.845
```

In [107]:
```
def predict(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
prediction = predict(y_test_pred,optimal_threshold)
```

In [108]:
```
from sklearn.metrics import confusion_matrix
matrix = confusion_matrix(y_test,prediction)
print('Confusion matrix : \n',matrix)
```

```
Confusion matrix :
 [[1446  868]
 [4544 8142]]
```

In [109]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.heatmap(matrix, annot=True,cmap='Blues',fmt='g')
plt.xlabel('Predicted label')
plt.ylabel('Actual label')
plt.title('Confusion matrix')
plt.show()
```



conclusion

In [111]:
```
#http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer","Model", "n_estimators", "Max Dept", "Test AUC"]

x.add_row(["TF-IDF", "GradientBoostingClassifier", 200, 5,0.69861])
x.add_row(["TF-IDF W2V", "GradientBoostingClassifier",100,5,0.6795])


print(x)
```

```
+------------+---------------------------+--------------+----------+----------+
| Vectorizer |           Model           | n_estimators | Max Dept | Test AUC |
+------------+---------------------------+--------------+----------+----------+
|   TF-IDF   | GradientBoostingClassifier |     200      |    5     | 0.69861  |
| TF-IDF W2V | GradientBoostingClassifier |     100      |    5     |  0.6795  |
+------------+---------------------------+--------------+----------+----------+
```

In [ ]:
```
!jupyter nbconvert --to html GBDT_solve.ipynb
```