```
 1
 2 %matplotlib inline
 3 import warnings
 4 warnings.filterwarnings("ignore")
 5
 6 import sqlite3
 7 import pandas as pd
 8 import numpy as np
 9 import nltk
10 import string
11 import matplotlib.pyplot as plt
12 import seaborn as sns
13 from sklearn.feature_extraction.text import TfidfTransformer
14 from sklearn.feature_extraction.text import TfidfVectorizer
15
16 from sklearn.feature_extraction.text import CountVectorizer
17 from sklearn.metrics import confusion_matrix
18 from sklearn import metrics
19 from sklearn.metrics import roc_curve, auc
20 from nltk.stem.porter import PorterStemmer
21
22 import re
23 import string
24 from nltk.corpus import stopwords
25 from nltk.stem import PorterStemmer
26 from nltk.stem.wordnet import WordNetLemmatizer
27
28 from gensim.models import Word2Vec
29 from gensim.models import KeyedVectors
30 import pickle
31
32 from tqdm import tqdm
33 import os
34
35 # from plotly import plotly
36 # import plotly.offline as offline
37 # import plotly.graph_objs as go
38 # offline.init_notebook_mode()
```

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

```
#getting the file from google drive (resources data)
import gdown

url = 'https://drive.google.com/uc?id=1OcMV5zjAJI7OvNxxN4Ant52BDF3jrZOZ'
output = 'resources.csv'
# https://drive.google.com/file/d/1OcMV5zjAJI7OvNxxN4Ant52BDF3jrZOZ/view?usp=sharing
gdown.download(url, output, quiet=False)

```

```
Downloading...
From: https://drive.google.com/uc?id=1OcMV5zjAJI7OvNxxN4Ant52BDF3jrZOZ
To: /content/resources.csv
127MB [00:00, 295MB/s]
'resources.csv'
```

```
#getting the data from google drive (test data)
import gdown

url = 'https://drive.google.com/uc?id=1JGtsNLea4Q2HZQIgBp3pRrOfRN80qIg0'
# https://drive.google.com/file/d/1JGtsNLea4Q2HZQIgBp3pRrOfRN80qIg0/view?usp=sharing
output = 'train_data.csv'
gdown.download(url, output, quiet=False)
```

```
Downloading...
From: https://drive.google.com/uc?id=1JGtsNLea4Q2HZQIgBp3pRrOfRN80qIg0
To: /content/train_data.csv
201MB [00:00, 246MB/s]
'train_data.csv'
```

```
ls
```

```
resources.csv  sample_data/  train_data.csv
```

```
Project_data = pd.read_csv("train_data.csv")
Resources_data = pd.read_csv("resources.csv")
print("Shape of Train data ",Project_data.shape)
```

```
print("Shape of Train data ",Resources_data.shape)
print("-"*100)
print("The attributes in Resources_data are as folows " , "  -> "*5,Resources_data.columns.values)
print("The attributes in Project_data are as folows " , "  ->  "*5,Project_data.columns.values)
```

```
Shape of Train data  (109248, 17)
Shape of Train data  (1541272, 4)
----------------------------------------------------------------------------------------------------
The attributes in Resources_data are as folows    ->   ->   ->   ->   ->  ['id' 'description' 'quantity' 'price']
The attributes in Project_data are as folows    ->    ->    ->    ->    ->   ['Unnamed: 0' 'id' 'teacher_id' 'teacher_p
 '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']
```

```
# here the code is chainging the date and time format used in the data to a standard form  and replacing the data with the
# replacing the same coloum  refrence to same are given below
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sort_values.html

Project_data["Date"] = pd.to_datetime(Project_data["project_submitted_datetime"])
Project_data.drop("project_submitted_datetime",axis=1,inplace = True )
Project_data.sort_values(by=["Date"],inplace=True)
Project_data.head(2)

```

```
Unnamed:
```

```
# find out the NAN values in the dataframe and fill with null
# https://stackoverflow.com/questions/29530232/how-to-check-if-any-value-is-nan-in-a-pandas-dataframe
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.fillna.html
Project_data["teacher_prefix"] = Project_data["teacher_prefix"].fillna("null")
print("The number of NAN values in teacher_prefix Column is " + str(Project_data["teacher_prefix"].isnull().sum()))
```

```
The number of NAN values in teacher_prefix Column is 0
```

```
76127  37728  p043609  3f60494c61921b3b43ab61bdde2904df  Ms.  UT  Grades 3-5
```

1.1 ** Preprocessing of the Text **

```
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

catogories = list(Project_data['project_subject_categories'].values)
cat_list = []
for i in catogories:
    temp = ""
    for j in i.split(','):
        if 'The' in j.split():
            j=j.replace('The','')
        j = j.replace(' ','')
        temp+=j.strip()+" "
        temp = temp.replace('&','_')
    cat_list.append(temp.strip().lower())

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

from collections import Counter
my_counter = Counter()
for word in Project_data['clean_categories'].values:
    my_counter.update(word.split())

```

```
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
print("The Worlds in sorted_cat_dict",sorted_cat_dict)
```

```
catogories1 = list(Project_data['project_subject_subcategories'].values)
cat_list1 = []
for i in catogories1:
    temp1 = ""
    for j in i.split(','):
        if 'The' in j.split():
            j=j.replace('The','')
        j = j.replace(' ','')
        temp1+=j.strip()+" "
        temp1 = temp1.replace('&','_')
    cat_list1.append(temp1.strip().lower())

Project_data['clean_sub_categories'] = cat_list1
Project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

from collections import Counter
my_counter1 = Counter()
for word in Project_data['clean_sub_categories'].values:
    my_counter1.update(word.split())

cat_dict1 = dict(my_counter1)
sorted_cat_dict1 = dict(sorted(cat_dict1.items(), key=lambda kv: kv[1]))
print("The Worlds in sorted_cat_dict1",sorted_cat_dict1)
for i in sorted_cat_dict1:
  print(i,sorted_cat_dict1[i])
```

```
school_state = list(Project_data['school_state'].values)
school_state_list = []
for i in school_state:
    temp2 = ""
    for j in i.split(','):
        if 'The' in j.split():
            j=j.replace('The','')
        j = j.replace(' ','')
        temp2 +=j.strip()+" "
        temp2 = temp2.replace('&','_')
    school_state_list.append(temp2.strip().lower())

Project_data['School_state'] = school_state_list
Project_data.drop(['school_state'], axis=1, inplace=True)
```

```
15
16 my_counter3 = Counter()
17 for word in Project_data['School_state'].values:
18     my_counter3.update(word.split())
19
20 school_state_dict = dict(my_counter3)
21 sorted_school_state_dict = dict(sorted(school_state_dict.items(), key=lambda kv: kv[1]))
22 print("The Values in sorted_school_state_dict : ", sorted_school_state_dict)
23
24
```

```
The Values in sorted_school_state_dict :  {'vt': 80, 'wy': 98, 'nd': 143, 'mt': 245, 'ri': 285, 'sd': 300, 'ne': 309, '
```

```
1 # merge two column text dataframe:
2 Project_data["essay"] = Project_data["project_essay_1"].map(str) +\
3                         Project_data["project_essay_2"].map(str) + \
4                         Project_data["project_essay_3"].map(str) + \
5                         Project_data["project_essay_4"].map(str)
```

```
1 # https://stackoverflow.com/a/47091490/4084039
2 import re
3
4 def decontracted(phrase):
5     # specific
6     phrase = re.sub(r"won't", "will not", phrase)
7     phrase = re.sub(r"can\'t", "can not", phrase)
8
9     # general
10    phrase = re.sub(r"n\'t", " not", phrase)
11    phrase = re.sub(r"\'re", " are", phrase)
12    phrase = re.sub(r"\'s", " is", phrase)
13    phrase = re.sub(r"\'d", " would", phrase)
14    phrase = re.sub(r"\'ll", " will", phrase)
15    phrase = re.sub(r"\'t", " not", phrase)
16    phrase = re.sub(r"\'ve", " have", phrase)
17    phrase = re.sub(r"\'m", " am", phrase)
18    return phrase
```

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', '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', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            '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', "weren't",
            'won', "won't", 'wouldn', "wouldn't"]
```

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

```
100%|██████████| 109248/109248 [01:01<00:00, 1788.74it/s]
```

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles = []
```

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

```
100%|██████████| 109248/109248 [00:02<00:00, 41717.06it/s]
```

```
Project_data["clean_titles"] = preprocessed_titles
Project_data.drop(["project_essay_1"],axis=1,inplace=True)
Project_data.drop(["project_essay_2"],axis=1,inplace=True)
Project_data.drop(["project_essay_3"],axis=1,inplace=True)
Project_data.drop(["project_essay_4"],axis=1,inplace=True)
Project_data.drop(["project_title"],axis=1,inplace=True)
```

Merging the dataframes that is Train_csv and Resources_csv

```
print("Project_data Columns are - >>> " , Project_data.columns ,"Resources_data Columns  ", Resources_data.columns)
```

```
Project_data Columns are - >>>  Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix',
       'project_grade_category', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'Date', 'clean_categories', 'clean_sub_categories', 'School_state',
       'essay', 'clean_titles'],
      dtype='object') Resources_data Columns   Index(['id', 'description', 'quantity', 'price'], dtype='object')
```

```
Price_data = Resources_data.groupby("id").agg({"price": "sum" ,"quantity" : "sum"}).reset_index()
Price_data.head(2)
```

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

```
1 Project_data = pd.merge(Project_data,Price_data,on="id",how = "left")
2 print("The Number of NAN values in PRoject_data : ",Project_data.isnull().sum())
```

```
The Number of NAN values in PRoject_data :  Unnamed: 0                                        0
id                                                0
teacher_id                                        0
teacher_prefix                                    0
project_grade_category                            0
project_resource_summary                          0
teacher_number_of_previously_posted_projects      0
project_is_approved                               0
Date                                              0
clean_categories                                  0
clean_sub_categories                              0
School_state                                      0
essay                                             0
clean_titles                                      0
price                                             0
quantity                                          0
dtype: int64
```

```
1 # Project_data = Project_data.head(50000)
2 Y = Project_data["project_is_approved"].values
3 X = Project_data.drop(["project_is_approved"],axis = 1)
4 print(X.columns)
5 print(Project_data.shape)
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix',
       'project_grade_category', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'Date',
       'clean_categories', 'clean_sub_categories', 'School_state', 'essay',
       'clean_titles', 'price', 'quantity'],
      dtype='object')
(109248, 16)
```

```
#Splitting the data into train and test data_set
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, stratify= Project_data['project_is_approved'],r
X_train, X_cv, Y_train, Y_cv = train_test_split(X_train, Y_train, test_size=0.33, stratify=Y_train)
```

```
1
```

```
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
import warnings
warnings.filterwarnings("ignore")

ros = RandomOverSampler(sampling_strategy='minority',random_state=42)
x_train, y_train = ros.fit_resample(X_train, Y_train)
print('Resampled dataset shape %s' % Counter(y_train))
print("Capitial" ,"X","represents the original train_data and lower case" ,"x", "represnts the ramdonly over-sampled data
```

```
Resampled dataset shape Counter({1: 41615, 0: 41615})
Capitial X represents the original train_data and lower case x represnts the ramdonly over-sampled data
```

```
# here we have to convert x into a dataframe
x_train = pd.DataFrame(x_train,columns = X.columns)
x_train.shape
# s= Project_data['project_is_approved'].value_counts()
# s

```

```
(83230, 15)
```

```
print("Shape of Train data ",x_train.shape, y_train.shape)
print("="*100)
print("Shape of Train CV  data ",X_cv.shape, Y_cv.shape)
print("="*100)
print("Shape of Test data ",X_test.shape, Y_test.shape)
print("="*100)
```

```
Shape of Train data  (83230, 15) (83230,)
====================================================================================================
Shape of Train CV  data  (24155, 15) (24155,)
====================================================================================================
Shape of Test data  (36052, 15) (36052,)
====================================================================================================
```

1.2 Vectorizing data

```
# One-Hotencoding for the Catogorial Values
# Standazing the Numerical Data for sclaing them to equal sclae
```

```
vectorizer1 = CountVectorizer()
vectorizer1.fit(x_train['clean_sub_categories']) # fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
x_train_clean_subcat_ohe = vectorizer1.transform(x_train['clean_sub_categories'])
X_cv_clean_subcat_ohe = vectorizer1.transform(X_cv['clean_sub_categories'])
X_test_clean_subcat_ohe = vectorizer1.transform(X_test['clean_sub_categories'])

print("After vectorizations of the clean_sub_categories , One-hot-encoding shape of the data is")
print("Shape of Train data ",x_train_clean_subcat_ohe.shape, y_train.shape)
print("Shape of Train CV  data ",X_cv_clean_subcat_ohe.shape, Y_cv.shape)
print("Shape of Test data " ,X_test_clean_subcat_ohe.shape, Y_test.shape)
```

```
vectorizer2 = CountVectorizer()
vectorizer2.fit(x_train['clean_categories']) # fit has to happen only on train data


# we use the fitted CountVectorizer to convert the text to vector
x_train_clean_categories_ohe = vectorizer2.transform(x_train['clean_categories'])
X_cv_clean_categories_ohe = vectorizer2.transform(X_cv['clean_categories'])
X_test_clean_categories_ohe = vectorizer2.transform(X_test['clean_categories'])

print("After vectorizations of the clean_categories, One-hot-encoding shape of the data is")
print("Shape of Train data ",x_train_clean_categories_ohe.shape, y_train.shape)
print("Shape of Train CV  data ",X_cv_clean_categories_ohe.shape, Y_cv.shape)
print("Shape of Test data " ,X_test_clean_categories_ohe.shape, Y_test.shape)
```

```
After vectorizations of the clean_categories, One-hot-encoding shape of the data is
Shape of Train data  (83230, 9) (83230,)
Shape of Train CV  data  (24155, 9) (24155,)
Shape of Test data  (36052, 9) (36052,)
```

```
vectorizer3 = CountVectorizer()
vectorizer3.fit(x_train['teacher_prefix'].values.astype('U')) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
x_train_teacher_ohe = vectorizer3.transform(x_train['teacher_prefix'].values.astype('U'))
X_cv_teacher_ohe = vectorizer3.transform(X_cv['teacher_prefix'].values.astype('U'))
X_test_teacher_ohe = vectorizer3.transform(X_test['teacher_prefix'].values.astype('U'))

print("After vectorizations of the teacher_prefix , One-hot-encoding shape of the data is")
print("Shape of Train data ",x_train_teacher_ohe.shape, y_train.shape)
print("Shape of Train CV  data ",X_cv_teacher_ohe.shape, Y_cv.shape)
print("Shape of Test data " ,X_test_teacher_ohe.shape, Y_test.shape)
```

```
After vectorizations of the teacher_prefix , One-hot-encoding shape of the data is
Shape of Train data  (83230, 6) (83230,)
Shape of Train CV  data  (24155, 6) (24155,)
Shape of Test data  (36052, 6) (36052,)
```

```
vectorizer4 = CountVectorizer()
```

```
vectorizer4.fit(x_train['School_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
x_train_state_ohe = vectorizer4.transform(x_train['School_state'].values)
X_cv_state_ohe = vectorizer4.transform(X_cv['School_state'].values)
X_test_state_ohe = vectorizer4.transform(X_test['School_state'].values)

print("After vectorizations of the School_state , One-hot-encoding shape of the data is")
print("Shape of Train data ",x_train_state_ohe.shape, y_train.shape)
print("Shape of Train CV  data ",X_cv_state_ohe.shape, Y_cv.shape)
print("Shape of Test data " ,X_test_state_ohe.shape, Y_test.shape)
```

```
After vectorizations of the School_state , One-hot-encoding shape of the data is
Shape of Train data  (83230, 51) (83230,)
Shape of Train CV  data  (24155, 51) (24155,)
Shape of Test data  (36052, 51) (36052,)
```

```
grade_cat_list = []
for grade in X_train['project_grade_category'].values:
  grade = grade.replace("-","_").lower()
  grade = grade.replace(" ","_").lower()
  grade_cat_list.append(grade)


X_train['clean_grade'] = grade_cat_list
X_train.drop(['project_grade_category'], axis=1, inplace=True)

my_counter = Counter()
for word in X_train['clean_grade'].values:
     my_counter.update(word.split())
project_grade_category_dict= dict(my_counter)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items(), key=lambda kv: kv[1]))
print(sorted_project_grade_category_dict)
```

```
{'grades_9_12': 4928, 'grades_6_8': 7534, 'grades_3_5': 16645, 'grades_prek_2': 19934}
```

```
vectorizer5 = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.keys()), lowercase=False, binary=True)
vectorizer5.fit(x_train['project_grade_category'].values) # fit has to happen only on train data
```

```
# we use the fitted CountVectorizer to convert the text to vector
x_train_grade_ohe = vectorizer5.transform(x_train['project_grade_category'].values)
X_cv_grade_ohe = vectorizer5.transform(X_cv['project_grade_category'].values)
X_test_grade_ohe = vectorizer5.transform(X_test['project_grade_category'].values)

print("After vectorizations of the project_grade_category , One-hot-encoding shape of the data is")
print("Shape of Train data ",x_train_grade_ohe.shape, y_train.shape)
print("Shape of Train CV  data ",X_cv_grade_ohe.shape, Y_cv.shape)
print("Shape of Test data " ,X_test_grade_ohe.shape,Y_test.shape)
```

```
After vectorizations of the project_grade_category , One-hot-encoding shape of the data is
Shape of Train data  (83230, 4) (83230,)
Shape of Train CV  data  (24155, 4) (24155,)
Shape of Test data  (36052, 4) (36052,)
```

Normalizing the numerical features: Price

```
from sklearn.preprocessing import Normalizer
standard_vector1 = Normalizer()
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshapstandard_vector2e(1, -1)  if it contains a single sample.
standard_vector1.fit(x_train['price'].values.reshape(-1,1))

x_train_price_std = standard_vector1.transform(x_train['price'].values.reshape(-1,1))
X_cv_price_std = standard_vector1.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_std = standard_vector1.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations of the price data ,  shape of the data after standazing")
print(x_train_price_std.shape, y_train.shape)
print(X_cv_price_std.shape, Y_cv.shape)
print(X_test_price_std.shape, Y_test.shape)
```

```
After vectorizations of the price data ,  shape of the data after standazing
(83230, 1) (83230,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

```
from sklearn.preprocessing import Normalizer
standard_vector2 = Normalizer()
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vector2.fit(x_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

x_train_projects_std = standard_vector2.transform(x_train['teacher_number_of_previously_posted_projects'].values.reshape(
X_cv_projects_std = standard_vector2.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_projects_std = standard_vector2.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1

print("After vectorizations of the teacher_number_of_previously_posted_projects , shape of the data after standazing")
print(x_train_projects_std.shape, y_train.shape)
print(X_cv_projects_std.shape,Y_cv.shape)
print(X_test_projects_std.shape, Y_test.shape)
```

```
After vectorizations of the teacher_number_of_previously_posted_projects , shape of the data after standazing
(83230, 1) (83230,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

```
from sklearn.preprocessing import Normalizer
standard_vector3 = Normalizer()
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
standard_vector3.fit(x_train['quantity'].values.reshape(-1,1))

x_train_qty_std = standard_vector3.transform(x_train['quantity'].values.reshape(-1,1))
```

```
x_train_qty_std = standard_vector3.transform(x_train['quantity'].values.reshape(-1,1))
X_cv_qty_std = standard_vector3.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_qty_std = standard_vector3.transform(X_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(x_train_qty_std.shape, y_train.shape)
print(X_cv_qty_std.shape, Y_cv.shape)
print(X_test_qty_std.shape, Y_test.shape)
```

```
After vectorizations
(83230, 1) (83230,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

2.1 Apply Multinomial NaiveBayes on these feature sets

Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)

Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

`**Difference between fit(),transform(),fit_transform()**`

To center the data (make it have zero mean and unit standard error), you subtract the mean and then divide the result by the standard deviation.

fit() just calculates the parameters (e.g. mu and sigma in case of StandardScaler) and saves them as an internal objects state.

Afterwards, you can call its transform() method to apply the transformation to a particular set of examples

for egs fit() function happens only on training data while transform () involves changing the values by keeping mu and sigma in calculation x'= ((x-mu)/sigma))

Using fix_transform(), we join these two steps and is used for the initial fitting of parameters on the training set x, but it also returns a transformed x’. Internally, it just calls first fit() and then transform() on the same data.

generally fit_transform() should be applied on train data,and not on cv and test data,once fit has been done then we can use transform () on cv and test data

2.1.1 Set : 1 Applying Naive Bayes on categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer6 = CountVectorizer(min_df=5,tokenizer = lambda x: x.split(), max_features=20000,ngram_range=(1, 4))
vectorizer6.fit(x_train['essay'].values) # fit has to happen only on train data

# we use the transform Text_data to vector , BOW CountVectorizer
x_train_essay_bow = vectorizer6.transform(x_train['essay'].values) # this the vectorization of the oversampled data we do
X_cv_essay_bow = vectorizer6.transform(X_cv['essay'].values)  # here we use the X_CV for vectorization( only x_train is o
X_test_essay_bow = vectorizer6.transform(X_test['essay'].values)

print("*"*100)
print("After vectorizations of the essay data the shape of the data is ")
print("Shape of the x_train data after Vectorization using BOW " , x_train_essay_bow.shape, y_train.shape)
print("Shape of the X_cv data after Vectorization using BOW " ,    X_cv_essay_bow.shape, Y_cv.shape)
print("Shape of the X_test_ data after Vectorization using BOW " ,    X_test_essay_bow.shape, Y_test.shape)
```

```
****************************************************************************************************
After vectorizations of the essay data the shape of the data is
Shape of the x_train data after Vectorization using BOW  (83230, 20000) (83230,)
Shape of the X_cv data after Vectorization using BOW  (24155, 20000) (24155,)
Shape of the X_test_ data after Vectorization using BOW  (36052, 20000) (36052,)
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer7 = CountVectorizer(min_df=5,tokenizer = lambda x: x.split(), max_features=5000)
vectorizer7.fit(x_train['clean_titles'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
x_train_titles_bow = vectorizer7.transform(x_train['clean_titles'].values)
X_cv_titles_bow = vectorizer7.transform(X_cv['clean_titles'].values)
X_test_titles_bow = vectorizer7.transform(X_test['clean_titles'].values)

print("After vectorizations of the clean_titles(Project Titles) data the shape of the data is")
print("Shape of the x_train data after Vectorization using BOW " ,x_train_titles_bow.shape, y_train.shape)
print("Shape of the X_cv data after Vectorization using BOW " , X_cv_titles_bow.shape, Y_cv.shape)
```

```
print("Shape of the X_cv data after Vectorization using BOW " ,X_cv_titles_bow.shape, Y_cv.shape)
print("Shape of the X_test_ data after Vectorization using BOW " ,X_test_titles_bow.shape, Y_test.shape)
```

```
After vectorizations of the clean_titles(Project Titles) data the shape of the data is
Shape of the x_train data after Vectorization using BOW  (83230, 4803) (83230,)
Shape of the X_cv data after Vectorization using BOW  (24155, 4803) (24155,)
Shape of the X_test_ data after Vectorization using BOW  (36052, 4803) (36052,)
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer8 = CountVectorizer(min_df=5,tokenizer = lambda x: x.split(), max_features=10000,ngram_range=(1, 4))
vectorizer8.fit(x_train['project_resource_summary']) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
x_train_summary_bow = vectorizer8.transform(x_train['project_resource_summary'])
X_cv_summary_bow = vectorizer8.transform(X_cv['project_resource_summary'])
X_test_summary_bow = vectorizer8.transform(X_test['project_resource_summary'])

print("After vectorizations of the project_resource_summary data the shape of the data is")
print("Shape of the x_train data after Vectorization using BOW " ,x_train_summary_bow.shape, y_train.shape)
print("Shape of the X_cv data after Vectorization using BOW " ,X_cv_summary_bow.shape, Y_cv.shape)
print("Shape of the X_test_ data after Vectorization using BOW " ,X_test_summary_bow.shape, Y_test.shape)
print("*"*100)
```

```
After vectorizations of the project_resource_summary data the shape of the data is
Shape of the x_train data after Vectorization using BOW  (83230, 10000) (83230,)
Shape of the X_cv data after Vectorization using BOW  (24155, 10000) (24155,)
Shape of the X_test_ data after Vectorization using BOW  (36052, 10000) (36052,)
****************************************************************************************************
```

Merging all the Vectorized data for making the Data matrix

```
from scipy.sparse import hstack
X1_tr = hstack((x_train_clean_categories_ohe,x_train_clean_subcat_ohe,x_train_teacher_ohe,x_train_state_ohe,\
                x_train_grade_ohe,x_train_titles_bow,x_train_essay_bow,x_train_price_std,x_train_projects_std,x_train_qty_
X1_cv = hstack((X_cv_clean_categories_ohe,X_cv_clean_subcat_ohe,X_cv_teacher_ohe,X_cv_state_ohe,X_cv_grade_ohe,\
                X_cv_price_std,X_cv_projects_std,X_cv_qty_std,X_cv_essay_bow,X_cv_titles_bow)).tocsr()
X1_te =hstack((X_test_clean_categories_ohe,X_test_clean_subcat_ohe,X_test_teacher_ohe,X_test_state_ohe,\
               X_test_grade_ohe,X_test_essay_bow,X_test_titles_bow,X_test_price_std,X_test_projects_std,X_test_qty_std)).
```

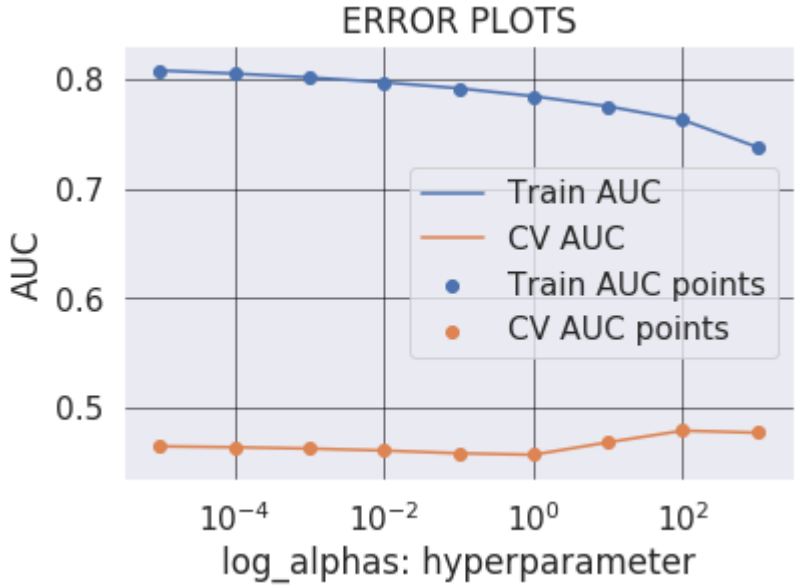
```

print("The final Data Matrix for Set:1" , " All the shapes of the data represent the merged features as mentioned in the t
print("shape of X_train is : ",            X1_tr.shape)
print("shape of X_Cross validation is :" , X1_cv.shape)
print("shape of X_test is ",               X1_te.shape)
```

```
The final Data Matrix for Set:1  All the shapes of the data represent the merged features as mentioned in the tittle
shape of X_train is :  (83230, 24906)
shape of X_Cross validation is : (24155, 24906)
shape of X_test is  (36052, 24906)
```

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
alpha = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10,  100,  1000]
for i in tqdm(alpha):
    neigh = MultinomialNB(alpha=i,class_prior=[0.5,0.5])
    neigh.fit(X1_tr, y_train)

    y_train_pred = neigh.predict_proba(X1_tr)[:,1]
    y_cv_pred = neigh.predict_proba(X1_cv)[:,1]

    # roc_auc_score(y_tr, y_score) the 2nd parameter should be probability estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(Y_cv, y_cv_pred))

plt.semilogx(alpha, train_auc, label='Train AUC')
plt.semilogx(alpha, cv_auc, label='CV AUC')

plt.scatter(alpha, train_auc, label='Train AUC points')
plt.scatter(alpha, cv_auc, label='CV AUC points')
print("Ploting the Log values of alpha as it would exactly plot the values of the point's considered ")
plt.legend()
```

```
plt.xlabel("log_alphas: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

```
100%|██████████| 9/9 [00:03<00:00,  2.64it/s]
Ploting the Log values of alpha as it would exactly plot the values of the point's considered
```

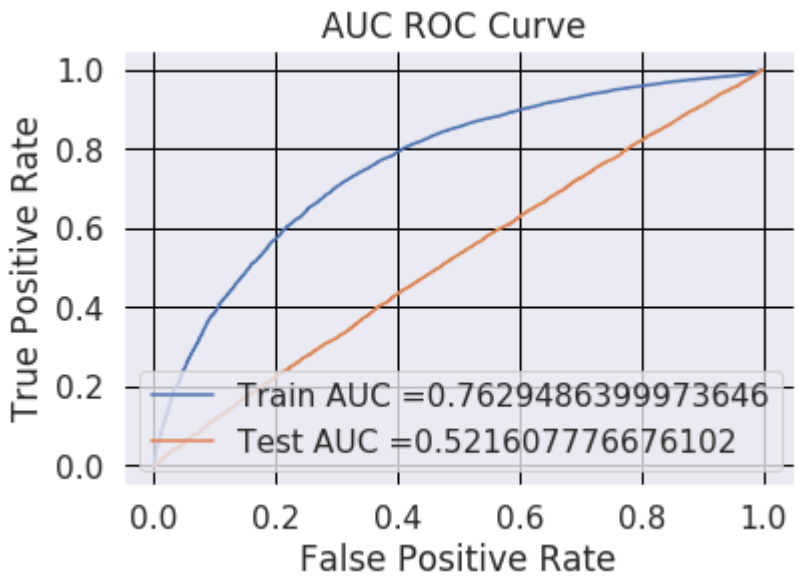



```
alpha1= 100
print("The Error Plot above shows the best Aplha value as :" , alpha )
```

```
The Error Plot above shows the best Aplha value as : [1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]
```

```
from sklearn.metrics import roc_curve, auc
neigh = MultinomialNB(alpha=alpha1,class_prior=[0.5,0.5])
neigh.fit(X1_tr, y_train)
y_train_pred = neigh.predict_proba(X1_tr)[:,1]
y_test_pred = neigh.predict_proba(X1_te)[:,1]
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
```

```
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("AUC ROC Curve ")
plt.grid(color='black', linestyle='-', linewidth=1)
plt.show()
```




```
#this is the custom function for predecting the best thershold and sorting the values according the threshould
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", np.round(max(tpr*(1-fpr)),3), "for threshold", np.round(t,3))
    return t
```

```
# we only set the threshould by using the train data and the test data is not altered at any time , as it could cause the
def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
```

```
        else:
            predictions.append(0)
    return predictions
```

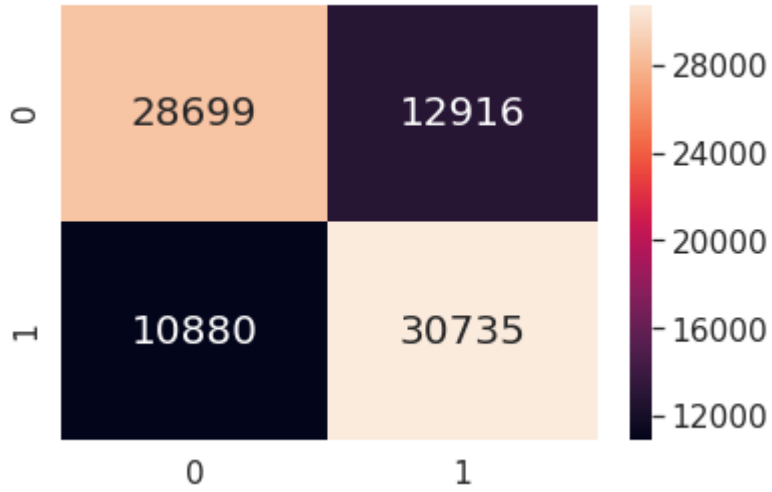
```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
# print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("tn, fp, fn, tp", "=", confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)).ravel())
print("Test confusion matrix")
# print(confusion_matrix(Y_test, predict_with_best_t(y_test_pred, best_t)))
print("tn, fp, fn, tp", "=", confusion_matrix(Y_test, predict_with_best_t(y_test_pred, best_t)).ravel())
print("here the threshold is ", np.round(best_t,3) , "i can change the threshold values according to Requirement in the c
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.509 for threshold 0.449
Train confusion matrix
tn, fp, fn, tp = [28699 12916 10880 30735]
Test confusion matrix
tn, fp, fn, tp = [ 2504  2955 13505 17088]
here the threshold is  0.449 i can change the threshold values according to Requirement in the confusion meterix
```

```
Confusion_metrix_Train_data = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
Confusion_metrix_Test_data = pd.DataFrame(confusion_matrix(Y_test, predict_with_best_t(y_test_pred, best_t)))

```

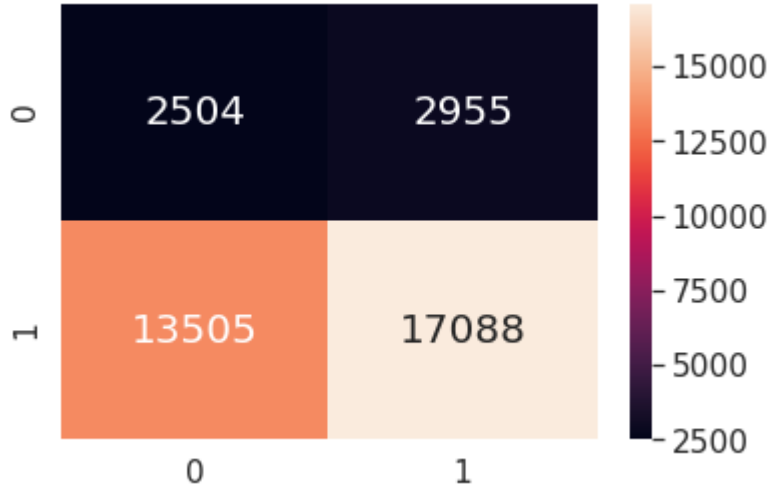
```
import seaborn as sns
sns.set(font_scale=1.4)#for label size
sns.heatmap(Confusion_metrix_Train_data,annot=True, annot_kws={"size": 20},fmt ="g")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4babd73a20>
```




```
1 sns.heatmap(Confusion_metrix_Test_data,annot=True, annot_kws={"size": 20},fmt ="g")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b8c1e05c0>
```



Conclusion for set 1:

1. The BOW for set 1 , gives us the results with a very low accuracy the AUC score was found out to be 50 which resembles that it is a random model a give random results when tested
2. The Confusion matrix gives us the TRP and FPR and it has been set with thershold of 0.509 , as I wanted to maximise the TPR with the help of settin thershold .
3. The model of BOW is not performing as expected beacuse it doesnt consider the least occuring words to be important it just creates a orthogonal ve the words which are present and dones not preserver the schematic meaning of the words too .

Area's of improvemnt's

1. The areas of imporvement are we can use Unigrams or "n -grams " this can imporove the models performance .

2.1.2 Set : 2 Categorical, Numerical features + Poject_title(TFIDF)+ Peprocessed_eassay (TFID

```
# Applying TF-IDF on Prohect title :
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer9 = TfidfVectorizer(min_df=10)
vectorizer9.fit(x_train["clean_titles"].values)

x_tain_project_titles_tfidf = vectorizer9.transform(x_train["clean_titles"].values)
X_cv_project_titles_tfidf = vectorizer9.transform(X_cv["clean_titles"].values)
X_test_project_titles_tfidf = vectorizer9.transform(X_test["clean_titles"].values)


print("After TFIDF vectorizations of the clean_titles , shape of the data after standazing")
print(x_tain_project_titles_tfidf.shape, y_train.shape)
print(X_cv_project_titles_tfidf.shape,Y_cv.shape)
print(X_test_project_titles_tfidf.shape, Y_test.shape)
print("*"*100)
```

```
After TFIDF vectorizations of the clean_titles , shape of the data after standazing
(83230, 3017) (83230,)
(24155, 3017) (24155,)
(36052, 3017) (36052,)
****************************************************************************************************
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer10 = TfidfVectorizer(min_df=10)
vectorizer10.fit(x_train["essay"])

x_tain_essay_tfidf = vectorizer10.transform(x_train["essay"].values)
X_cv_essay_tfidf = vectorizer10.transform(X_cv["essay"].values)
X_test_essay_tfidf = vectorizer10.transform(X_test["essay"].values)


print("After TFIDF vectorizations of the essay , shape of the data after standazing")
print(x_tain_essay_tfidf.shape, y_train.shape)
print(X_cv_essay_tfidf.shape,Y_cv.shape)
print(X_test_essay_tfidf.shape, Y_test.shape)
print("*"*100)
```

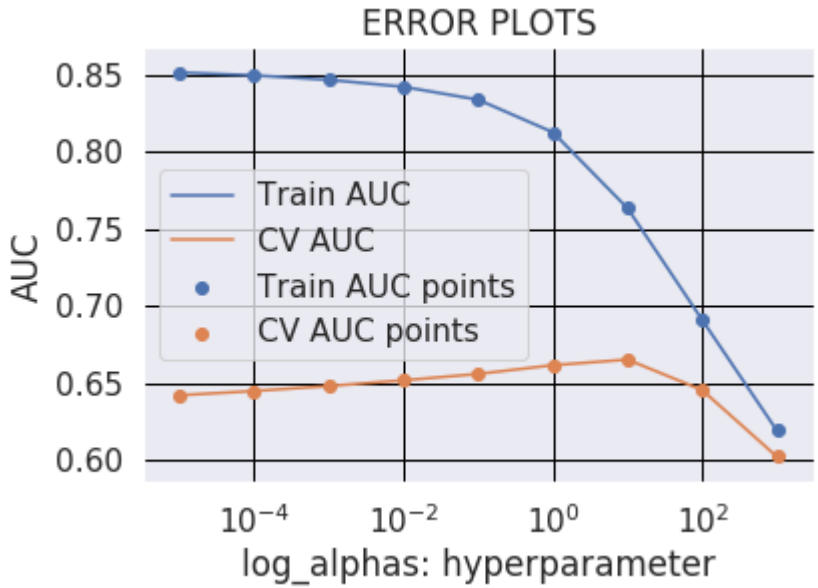
```
After TFIDF vectorizations of the essay , shape of the data after standazing
(83230, 15872) (83230,)
(24155, 15872) (24155,)
(36052, 15872) (36052,)
****************************************************************************************************
```

```
from scipy.sparse import hstack
X2_tr = hstack((x_train_clean_categories_ohe,x_train_clean_subcat_ohe,x_train_teacher_ohe,x_train_state_ohe,\
                x_train_grade_ohe,x_train_price_std,x_train_projects_std,x_train_qty_std,x_tain_project_titles_tfidf,x_ta
X2_cv = hstack((X_cv_clean_categories_ohe,X_cv_clean_subcat_ohe,X_cv_teacher_ohe,X_cv_state_ohe,X_cv_grade_ohe,\
                X_cv_price_std,X_cv_projects_std,X_cv_qty_std,X_cv_project_titles_tfidf,X_cv_essay_tfidf)).tocsr()
X2_te =hstack((X_test_clean_categories_ohe,X_test_clean_subcat_ohe,X_test_teacher_ohe,X_test_state_ohe,\
               X_test_grade_ohe,X_test_price_std,X_test_projects_std,X_test_qty_std,X_test_project_titles_tfidf,X_test_es


print("The final Data Matrix for Set:2" , " All the shapes of the data represent the merged features as mentioned in the
print("shape of X_train is : ",           X2_tr.shape)
print("shape of X_Cross validation is :" , X2_cv.shape)
print("shape of X_test is ",              X2_te.shape)
```

The final Data Matrix for Set:2 All the shapes of the data represent the merged features as mentioned in the tittle

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc1 = []
cv_auc1 = []
alpha = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10,  100,  1000]
for i in tqdm(alpha):
    neigh = MultinomialNB(alpha=i,class_prior=[0.5,0.5])
    neigh.fit(X2_tr, y_train)

      y_train_pred = neigh.predict_proba(X2_tr)[:,1]
    y_cv_pred = neigh.predict_proba(X2_cv)[:,1]

    # roc_auc_score(y_tr, y_score) the 2nd parameter should be probability estimates of the positive class
    # not the predicted outputs
    train_auc1.append(roc_auc_score(y_train,y_train_pred))
    cv_auc1.append(roc_auc_score(Y_cv, y_cv_pred))

plt.semilogx(alpha, train_auc1, label='Train AUC')
plt.semilogx(alpha, cv_auc1, label='CV AUC')

plt.scatter(alpha, train_auc1, label='Train AUC points')
plt.scatter(alpha, cv_auc1, label='CV AUC points')
print("Ploting the Log values of alpha as it would exactly plot the values of the point's considered ")
plt.legend()
plt.xlabel("log_alphas: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(color='black', linestyle='-', linewidth=1)
plt.show()
```

```
100%|██████████| 9/9 [00:01<00:00,  4.66it/s]
Ploting the Log values of alpha as it would exactly plot the values of the point's considered
```

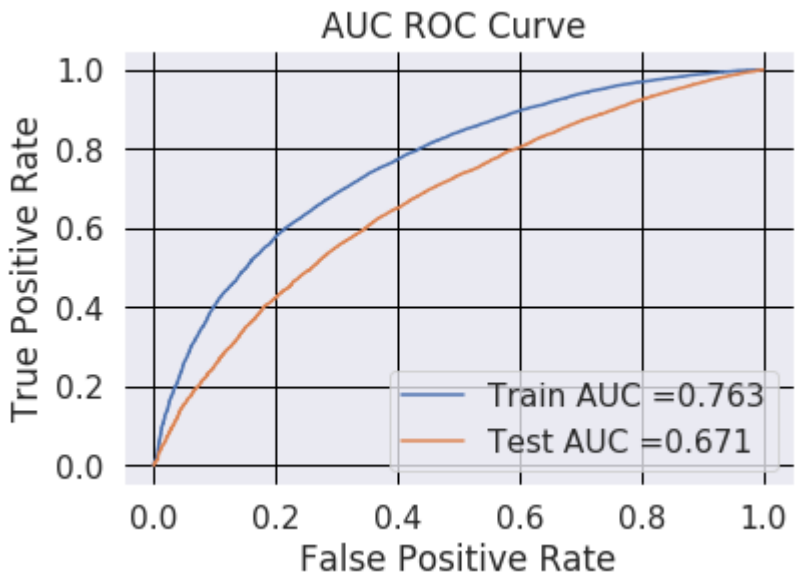



```
alpha= 10
print("The Error Plot above shows the best Aplha value as :" , alpha )
```

```
The Error Plot above shows the best Aplha value as : 10
```

```
from sklearn.metrics import roc_curve, auc
neigh = MultinomialNB(alpha=alpha,class_prior=[0.5,0.5])
neigh.fit(X2_tr, y_train)
y_train_pred1 = neigh.predict_proba(neigh, X2_tr)
y_test_pred1 = neigh.predict_proba(neigh, X2_te)
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred1)
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_test_pred1)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+ str(np.round(auc(train_fpr, train_tpr),3)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(np.round(auc(test_fpr, test_tpr),3)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("AUC ROC Curve ")
```

```
plt.grid(color='black', linestyle='-', linewidth=1)
plt.show()
```




```
#this is the custom function for predecting the best thershold and sorting the values according the threshould
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", np.round(max(tpr*(1-fpr)),3), "for threshold", np.round(t,3))
    return t
```

```
def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

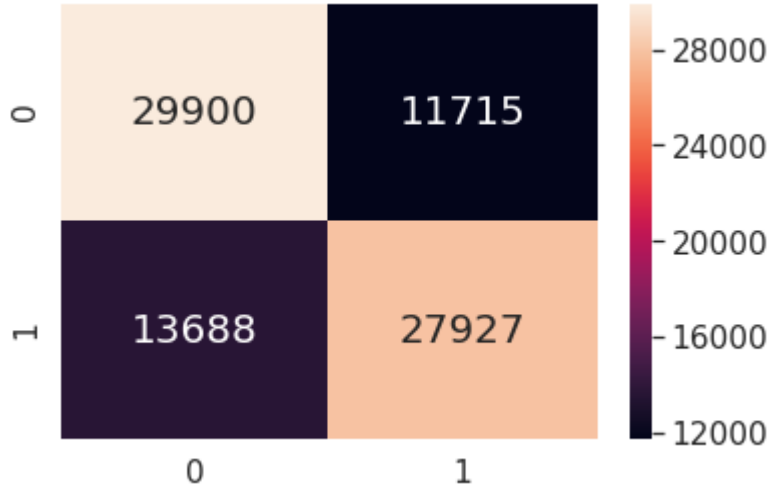
```
print("="*100)
from sklearn.metrics import confusion_matrix
```

```
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
# print(confusion_matrix(y_train, predict_with_best_t(y_train_pred1, best_t)))
print("tn, fp, fn, tp", "=", confusion_matrix(y_train, predict_with_best_t(y_train_pred1, best_t)).ravel())
print("Test confusion matrix")
# print(confusion_matrix(Y_test, predict_with_best_t(y_test_pred1, best_t)))
print("tn, fp, fn, tp", "=", confusion_matrix(Y_test, predict_with_best_t(y_test_pred1, best_t)).ravel())
print("here the threshold is ", np.round(best_t,3) , "i can change the threshold values according to Requirement in the c
```

```
==================================================================================================
the maximum value of tpr*(1-fpr) 0.482 for threshold 0.516
Train confusion matrix
tn, fp, fn, tp = [29900 11715 13688 27927]
Test confusion matrix
tn, fp, fn, tp = [ 3284  2175 10735 19858]
here the threshold is  0.516 i can change the threshold values according to Requirement in the confusion meterix
```

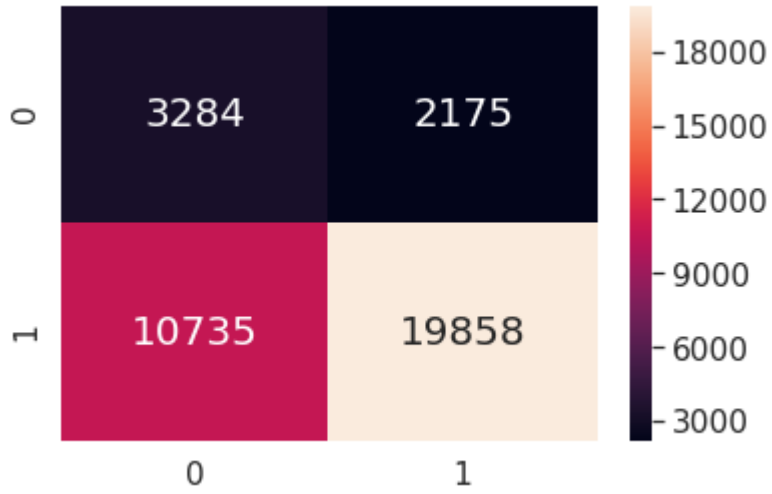
```
Confusion_metrix_Train_data = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_train_pred1, best_t)))
Confusion_metrix_Test_data = pd.DataFrame(confusion_matrix(Y_test, predict_with_best_t(y_test_pred1, best_t)))
import seaborn as sns
sns.set(font_scale=1.4)#for label size
print("Confusion metrix for Train data ")
sns.heatmap(Confusion_metrix_Train_data,annot=True, annot_kws={"size": 20},fmt ="g")
```

```
Confusion metrix for Train data
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b925f3780>
```




```
1 print("Confusion metrix for Test data")
2 sns.heatmap(Confusion_metrix_Test_data,annot=True, annot_kws={"size": 20},fmt ="g")
```

```
Confusion metrix for Test data
<matplotlib.axes._subplots.AxesSubplot at 0x7f4b82dd56d8>
```



```
## Conclusion for set 2:
```

1. TFIDF vectorizer gives very good results as compared to BOW as it equally considers the occurance of words in the Document Corpus
2. The AUC was found arounf 70 % and hance could be increased by using N-grams.
3. The confusion matrix was formed using the thershould of 0.49 for maximising the TPR.

3.1 Top 10 features of "positive class" and top 10 features of "negative class" for feature matrix Set 1

```
# Feature set Consitis of all the Catogorial and Numerical features of BOW vectorization

X1_tr = hstack((x_train_clean_categories_ohe,x_train_clean_subcat_ohe,x_train_teacher_ohe,x_train_state_ohe,\
                x_train_grade_ohe,x_train_titles_bow,x_train_essay_bow,x_train_price_std,x_train_projects_std,x_train_qty_
X1_cv = hstack((X_cv_clean_categories_ohe,X_cv_clean_subcat_ohe,X_cv_teacher_ohe,X_cv_state_ohe,X_cv_grade_ohe,\
                X_cv_price_std,X_cv_projects_std,X_cv_qty_std,X_cv_essay_bow,X_cv_titles_bow)).tocsr()
X1_te =hstack((X_test_clean_categories_ohe,X_test_clean_subcat_ohe,X_test_teacher_ohe,X_test_state_ohe,\
               X_test_grade_ohe,X_test_essay_bow,X_test_titles_bow,X_test_price_std,X_test_projects_std,X_test_qty_std)).t



print("The final Data Matrix for Set:1" , " All the shapes of the data represent the merged features as mentioned in the t
print("shape of X_train is : ",            X1_tr.shape)
print("shape of X_Cross validation is :" , X1_cv.shape)
print("shape of X_test is ",               X1_te.shape)
```

```
The final Data Matrix for Set:1  All the shapes of the data represent the merged features as mentioned in the tittle
shape of X_train is :  (83230, 24906)
shape of X_Cross validation is : (24155, 24906)
shape of X_test is  (36052, 24906)
```

```
nb_bow = MultinomialNB(alpha =alpha1 ,class_prior=[0.5,0.5])
nb_bow.fit(X1_tr, y_train)
```

```
MultinomialNB(alpha=100, class_prior=[0.5, 0.5], fit_prior=True)
```

```
neg_class_prob_sorted = nb_bow.feature_log_prob_[0, :].argsort()
pos_class_prob_sorted = nb_bow.feature_log_prob_[1, :].argsort()
```

```
# https://stackoverflow.com/questions/14131615/possible-to-append-multiple-lists-at-once-python
from itertools import chain
Stacked_Feature_list = list(chain(vectorizer1.get_feature_names(),vectorizer2.get_feature_names(),vectorizer3.get_feature_
                                  vectorizer4.get_feature_names(),vectorizer5.get_feature_names(),vectorizer6.get_feature_
                                  vectorizer7.get_feature_names(),vectorizer8.get_feature_names(),x_train_qty_std,x_train_


```

```
print("The words with higest importance in Postive class is")
print(np.take(Stacked_Feature_list, neg_class_prob_sorted[-30:-1]))
print("*"*20)
print("The words with higest importance in Negative class is")
print(np.take(Stacked_Feature_list, pos_class_prob_sorted[-30:-1]))
```

```
The words with higest importance in Postive class is
['my class and' 'fruits and vegetables' 'our school are' 'you in advance'
 'group of students who' 'students enter' 'enjoy' 'own' 'meaning'
 'will inspire' 'explore their' 'their communities' 'small'
 'school to learn' 'students will feel' 'middle class' 'of skills'
 'learners have' 'together they' 'radical' 'while having'
 'with your help we' 'donating to our project' 'school and my'
 'need to become' 'starting to' 'time or' 'up in the' 'comes']
********************
The words with higest importance in Negative class is
['on regular basis' 'our school are' 'fruits and vegetables'
 'you in advance' 'group of students who' 'students enter' 'enjoy' 'own'
 'will inspire' 'meaning' 'explore their' 'their communities' 'small'
 'of skills' 'middle class' 'together they' 'school to learn'
 'students will feel' 'learners have' 'radical' 'while having'
 'with your help we' 'donating to our project' 'school and my'
 'need to become' 'starting to' 'time or' 'up in the' 'comes']
```

3.2 Top 10 features of "positive class" and top 10 features of "negative class" for feature matrix Set 2

```
from scipy.sparse import hstack
X2_tr = hstack((x_train_clean_categories_ohe,x_train_clean_subcat_ohe,x_train_teacher_ohe,x_train_state_ohe,\
                x_train_grade_ohe,x_train_price_std,x_train_projects_std,x_train_qty_std,x_tain_project_titles_tfidf,x_ta
X2_cv = hstack((X_cv_clean_categories_ohe,X_cv_clean_subcat_ohe,X_cv_teacher_ohe,X_cv_state_ohe,X_cv_grade_ohe,\
                X_cv_price_std,X_cv_projects_std,X_cv_qty_std,X_cv_project_titles_tfidf,X_cv_essay_tfidf)).tocsr()
X2_te =hstack((X_test_clean_categories_ohe,X_test_clean_subcat_ohe,X_test_teacher_ohe,X_test_state_ohe,\
               X_test_grade_ohe,X_test_price_std,X_test_projects_std,X_test_qty_std,X_test_project_titles_tfidf,X_test_ess



print("The final Data Matrix for Set:2" , " All the shapes of the data represent the merged features as mentioned in the t
print("shape of X_train is : ",            X2_tr.shape)
print("shape of X_Cross validation is :" , X2_cv.shape)
print("shape of X_test is ",               X2_te.shape)
```

```
The final Data Matrix for Set:2  All the shapes of the data represent the merged features as mentioned in the tittle
shape of X_train is :  (83230, 18992)
shape of X_Cross validation is : (24155, 18992)
shape of X_test is  (36052, 18992)
```

```
nb_bow2 = MultinomialNB(alpha=alpha ,class_prior=[0.5,0.5])
nb_bow2.fit(X2_tr, y_train)
```

```
MultinomialNB(alpha=0.01, class_prior=[0.5, 0.5], fit_prior=True)
```

```
neg_class_prob_sorted_set2 = nb_bow2.feature_log_prob_[0, :].argsort()
pos_class_prob_sorted_set2 = nb_bow2.feature_log_prob_[1, :].argsort()
```

```
# https://stackoverflow.com/questions/14131615/possible-to-append-multiple-lists-at-once-python
from itertools import chain
Stacked_Feature_list1 = list(chain(vectorizer1.get_feature_names(),vectorizer2.get_feature_names(),vectorizer3.get_feature
                                   vectorizer4.get_feature_names(),vectorizer5.get_feature_names(),vectorizer9.get_feature_
                                   vectorizer10.get_feature_names(),x_train_qty_std,x_train_projects_std,x_train_price_std



# thers a import point to note that is if we take argsort then it gives values in ascending order so we have to select fe
print("The words with higest importance in Postive class is")
# i took features from -30 to -1 which means select all the features which are most important form the accendingly sorted
```

```
print(np.take(Stacked_Feature_list1, neg_class_prob_sorted[-30:-1]))
print("*"*20)
print("The words with higest importance in Negative class is")
print(np.take(Stacked_Feature_list1, pos_class_prob_sorted[-30:-1]))
```

```
The words with higest importance in Postive class is
['influence' 'cooler' 'mixing' array([1.]) 'dealing' 'provoking' 'cameras'
 'movable' array([1.]) array([1.]) 'children' 'slime' array([1.]) 'oxygen'
 'receivership' 'identical' 'looming' 'furthers' 'utilized' array([1.])
 array([1.]) array([1.]) 'blog' 'organization' 'jamestown' 'poured'
 'taylor' 'vitamin' 'ankle']
********************
The words with higest importance in Negative class is
['march' 'mixing' 'cooler' array([1.]) 'dealing' 'provoking' 'cameras'
 'movable' array([1.]) array([1.]) 'children' 'slime' array([1.])
 'looming' 'identical' 'utilized' 'oxygen' 'receivership' 'furthers'
 array([1.]) array([1.]) array([1.]) 'blog' 'organization' 'jamestown'
 'poured' 'taylor' 'vitamin' 'ankle']
```

Conclusion

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Alpha:Hyper Parameter", " Test AUC"]
x.add_row(["BOW", "Multinomial Naive Bayes", 0.01, 0.55])
x.add_row(["TFIDF", "Multinomial Naive Bayes", 0.01, 0.68])
print(x)
```

```
+------------+------------------------+----------------------+-----------+
| Vectorizer |          Model         | Alpha:Hyper Parameter |  Test AUC |
+------------+------------------------+----------------------+-----------+
|    BOW     | Multinomial Naive Bayes |         0.01         |    0.55   |
|   TFIDF    | Multinomial Naive Bayes |         0.01         |    0.68   |
+------------+------------------------+----------------------+-----------+
```