```
!pip install chart_studio
```

```
Collecting chart_studio
  Downloading chart_studio-1.1.0-py3-none-any.whl (64 kB)
     |████████████████████████████████| 64 kB 2.0 MB/s
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from chart_studio
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from chart_stu
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from chart_s
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from req
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (fr
Installing collected packages: chart-studio
Successfully installed chart-studio-1.1.0
```

```
%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 sklearn.model_selection import train_test_split
from sklearn.preprocessing import Normalizer
from scipy.sparse import hstack
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
import math

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/

import pickle
from tqdm import tqdm
import os

from chart_studio import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

1. Loading Data

```
os.chdir("/content/drive/MyDrive/Datasets/Assignment 8 Apply Naive Bayes on Donors Choose dataset")

train_data=pd.read_csv("train_data.csv",nrows=50000)
resources=pd.read_csv("resources.csv")

train_data.shape
```

```
(50000, 17)
```

```
resources.shape
```

```
(1541272, 4)
```

```
train_data.head()
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_s
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

```
resources.head()
```

2. Preprocessing

2.1 feature: project_grade_category

operations:

1. replace the spaces with '_'
2. replace the '-' with '_'
3. convert to lowercase

```
#pgrade=train_data["project_grade_category"].values  #if we assign without .values the type of pgrad
train_data["project_grade_category"]= train_data["project_grade_category"].apply(lambda x: x.replace
train_data["project_grade_category"]= train_data["project_grade_category"].str.replace('-','_')
train_data["project_grade_category"]= train_data["project_grade_category"].str.lower()
```

```
train_data["project_grade_category"]
```

```
0        grades_prek_2
1           grades_6_8
2           grades_6_8
3        grades_prek_2
4        grades_prek_2
             ...
49995    grades_prek_2
49996       grades_3_5
49997       grades_3_5
49998    grades_prek_2
49999    grades_prek_2
Name: project_grade_category, Length: 50000, dtype: object
```

2.2 feature: project_subject_categories

operations:

1. remove spaces and 'the'
2. replace '&' with '', *and* ',' *with* ''

```
train_data["project_subject_categories"].value_counts()
```

```
Literacy & Language                          10927
Math & Science                                7695
Literacy & Language, Math & Science           6705
Health & Sports                               4700
Music & The Arts                              2358
Special Needs                                 1913
Literacy & Language, Special Needs            1814
Applied Learning                              1719
Math & Science, Literacy & Language           1041
Applied Learning, Literacy & Language         1018
Math & Science, Special Needs                  871
History & Civics                               839
Literacy & Language, Music & The Arts          794
Math & Science, Music & The Arts               755
```

```
Applied Learning, Special Needs                672
History & Civics, Literacy & Language          651
Health & Sports, Special Needs                 633
Warmth, Care & Hunger                          606
Math & Science, Applied Learning               565
Applied Learning, Math & Science               477
Health & Sports, Literacy & Language           369
Literacy & Language, History & Civics          363
Applied Learning, Music & The Arts             360
Math & Science, History & Civics               282
Literacy & Language, Applied Learning          280
Applied Learning, Health & Sports              264
Math & Science, Health & Sports                187
History & Civics, Math & Science               171
Special Needs, Music & The Arts                140
History & Civics, Music & The Arts             135
Health & Sports, Math & Science                118
History & Civics, Special Needs                103
Health & Sports, Applied Learning               99
Applied Learning, History & Civics              78
Music & The Arts, Special Needs                 67
Health & Sports, Music & The Arts               66
Literacy & Language, Health & Sports            33
History & Civics, Applied Learning              25
Health & Sports, History & Civics               25
Special Needs, Health & Sports                  14
Health & Sports, Warmth, Care & Hunger          12
Music & The Arts, Health & Sports               10
Music & The Arts, History & Civics               9
History & Civics, Health & Sports                8
Applied Learning, Warmth, Care & Hunger          8
Math & Science, Warmth, Care & Hunger            7
Special Needs, Warmth, Care & Hunger             6
Music & The Arts, Applied Learning               4
Literacy & Language, Warmth, Care & Hunger       3
Music & The Arts, Warmth, Care & Hunger          1
Name: project_subject_categories, dtype: int64
```

```
train_data['project_subject_categories']=train_data['project_subject_categories'].str.replace(' the'
```

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

```
literacy_language                       10927
math_science                             7695
literacy_language_math_science           6705
health_sports                            4700
music_thearts                            2358
specialneeds                             1913
literacy_language_specialneeds           1814
appliedlearning                          1719
math_science_literacy_language           1041
appliedlearning_literacy_language        1018
math_science_specialneeds                 871
history_civics                            839
literacy_language_music_thearts           794
math_science_music_thearts                755
appliedlearning_specialneeds              672
history_civics_literacy_language          651
health_sports_specialneeds                633
warmth_care_hunger                        606
math_science_appliedlearning              565
appliedlearning_math_science              477
```

```
health_sports_literacy_language           369
literacy_language_history_civics          363
appliedlearning_music_thearts             360
math_science_history_civics               282
literacy_language_appliedlearning         280
appliedlearning_health_sports             264
math_science_health_sports                187
history_civics_math_science               171
specialneeds_music_thearts                140
history_civics_music_thearts              135
health_sports_math_science                118
history_civics_specialneeds               103
health_sports_appliedlearning              99
appliedlearning_history_civics             78
music_thearts_specialneeds                 67
health_sports_music_thearts                66
literacy_language_health_sports            33
health_sports_history_civics               25
history_civics_appliedlearning             25
specialneeds_health_sports                 14
health_sports_warmth_care_hunger           12
music_thearts_health_sports                10
music_thearts_history_civics                9
history_civics_health_sports                8
appliedlearning_warmth_care_hunger          8
math_science_warmth_care_hunger             7
specialneeds_warmth_care_hunger             6
music_thearts_appliedlearning               4
literacy_language_warmth_care_hunger        3
music_thearts_warmth_care_hunger            1
Name: project_subject_categories, dtype: int64
```

2.3 feature: project_subject_subcategories

operations: same as project_subject_categories

```
train_data['project_subject_subcategories']=train_data['project_subject_subcategories'].str.replace(
```

```
train_data["project_subject_subcategories"].value_counts()
```

```
literacy                                4434
literacy_mathematics                    3833
literature_writing_mathematics          2705
literacy_literature_writing             2570
mathematics                             2441
                                        ...
economics_nutritioneducation               1
civics_government_extracurricular          1
communityservice_financialliteracy         1
communityservice_music                     1
environmentalscience_financialliteracy     1
Name: project_subject_subcategories, Length: 384, dtype: int64
```

2.4 feature: teacher_prefix

operations:

1. Replace Nan with 'Mrs'
2. Remove the trailing '.'
3. convert to lower

```
train_data["teacher_prefix"].isnull().value_counts()
```

```
False    49998
True         2
Name: teacher_prefix, dtype: int64
```

```
train_data["teacher_prefix"]=train_data["teacher_prefix"].fillna('Mrs.')
```

```
train_data["teacher_prefix"].isnull().value_counts()
```

```
False    50000
Name: teacher_prefix, dtype: int64
```

```
train_data["teacher_prefix"]=train_data["teacher_prefix"].str.replace('.','').str.lower()
train_data["teacher_prefix"].value_counts()
```

```
mrs        26142
ms         17936
mr          4859
teacher     1061
dr             2
Name: teacher_prefix, dtype: int64
```

2.5 feature: school_state

operations: convert to lower

```
train_data["school_state"]=train_data["school_state"].str.lower()
```

```
train_data["school_state"].value_counts()
```

```
ca    7024
ny    3393
tx    3320
fl    2839
nc    2340
il    1967
sc    1830
ga    1828
mi    1468
pa    1419
oh    1180
in    1171
mo    1166
wa    1103
la    1094
ma    1076
ok    1074
nj    1005
az     994
va     916
wi     833
ut     792
al     790
ct     774
tn     774
md     668
```

```
nv     665
ky     614
ms     598
or     577
mn     556
co     538
ar     446
ia     306
id     302
ks     285
dc     247
hi     239
nm     236
me     222
wv     218
de     155
ak     153
ne     144
sd     142
nh     141
ri     126
mt     106
nd      63
wy      51
vt      32
Name: school_state, dtype: int64
```

2.6 feature: project_title

operations:

1. replacing english contractions by their actual meaning; can't-> can not etc.
2. stopwords removal
3. convert to lower case

```
# 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', 'himse
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'thes
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'w
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', '
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', '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'
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn'
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't
            'won', "won't", 'wouldn', "wouldn't"]


# https://stackoverflow.com/a/47091490/4084039
import re

def remove_contraction(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
```

```
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 = remove_contraction(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

preprocessed_titles = preprocess_text(train_data['project_title'].values)
train_data['project_title']=preprocessed_titles
train_data['project_title'].head
```

```
 100%|██████████| 50000/50000 [00:01<00:00, 29569.88it/s]
 <bound method NDFrame.head of 0              educational support english learners home
 1                      wanted projector hungry learners
 2        soccer equipment awesome middle school students
 3                                techie kindergarteners
 4                                interactive math tools
                             ...
 49995                   iteach using ipads instruction
 49996                     starbucks classroom redesign
 49997                       active bodies active minds
 49998                                      read writing
 49999                   inspiring young authors reading
 Name: project_title, Length: 50000, dtype: object>
```

2.7 feature: Essay

Operations:

1. Combine 'em all 'project_essay_1', 'project_essay_2', 'project_essay_3', 'project_essay_4'
2. replacing english contractions by their actual meaning; can't-> can not etc.
3. stopwords removal
4. convert to lower case

```
#it'll create a new feature essay in train_data which is a concatenation of all 4 essay categories
train_data["essay"] = train_data["project_essay_1"].map(str) +\
                      train_data["project_essay_2"].map(str) + \
                      train_data["project_essay_3"].map(str) + \
                      train_data["project_essay_4"].map(str)
```

```
preprocessed_essays = preprocess_text(train_data['essay'].values)
```

```
100%|██████████| 50000/50000 [00:38<00:00, 1298.37it/s]
```

```
train_data["essay"]=preprocessed_essays
```

```
train_data["essay"].values[1]
```

```
'students arrive school eager learn polite generous strive best know education succeed life he
lp improve lives school focuses families low incomes tries give student education deserve not
much students use materials given best projector need school crucial academic improvement stud
ents technology continues grow many resources internet teachers use growth students however sc
hool limited resources particularly technology without disadvantage one things could really he
lp classrooms projector projector not crucial instruction also growth students projector show
```

2.8 feature: price

```
resources.head()
```

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

```
#group by id; adding the price of all the entries having same id and similarly adding quantity to ob
price_data=resources.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

```
price_data["total_cost"]=price_data["price"]*price_data["quantity"]
```

```
price_data.head(2)
```

	id	price	quantity	total_cost
0	p000001	459.56	7	3216.92
1	p000002	515.89	21	10833.69

```
price_data=price_data.drop(["price","quantity"],axis=1)
```

```
price_data.head(2)
```

	id	total_cost
0	p000001	3216.92
1	p000002	10833.69

```
train_data=pd.merge(train_data, price_data, on='id',how='left')
```

```
train_data['total_cost'].head()
```

```
0     3555.80
1      299.00
2    11370.70
3      931.60
4      271.92
Name: total_cost, dtype: float64
```

```
train_data.shape
```

```
(50000, 19)
```

Double-click (or enter) to edit

Train Test Split

```
y=train_data['project_is_approved'].values
x=train_data.drop(['project_is_approved'],axis=1)
```

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, stratify=y)
```

3. Vectorization

3.1 Text data: Essay

```
#bag of word vectors
# min_df=10 specifies that for a word to be considered as a feature it has to occur in atleast 10 do
vec_essay_bow = CountVectorizer(min_df=10,max_features=5000)
vec_essay_bow.fit(x_train['essay'])
```

```
essay_bow_train = vec_essay_bow.transform(x_train['essay'])
print(essay_bow_train.shape)
```

```
(33500, 5000)
```

```
#test essays(bow)
essay_bow_test = vec_essay_bow.transform(x_test['essay'])
print(essay_bow_test.shape)
```

```
(16500, 5000)
```

```
#tfidf of essay
vec_tfidf_essay = TfidfVectorizer(min_df=10,max_features=5000)
vec_tfidf_essay.fit(x_train['essay'])

essay_tfidf_train = vec_tfidf_essay.transform(x_train['essay'])
essay_tfidf_test = vec_tfidf_essay.transform(x_test['essay'])

print("test ",essay_tfidf_test.shape)
print("train ",essay_tfidf_train.shape)
```

```
test  (16500, 5000)
train  (33500, 5000)
```

3.2 Text data- project_title

```
#bag of word vectors

vec_title_bow = CountVectorizer(min_df=10,max_features=5000)
vec_title_bow.fit(x_train['project_title'])
title_bow_train = vec_title_bow.transform(x_train['project_title'])
title_bow_test = vec_title_bow.transform(x_test['project_title'])
print(title_bow_test.shape)
print(title_bow_train.shape)
```

```
(16500, 1562)
(33500, 1562)
```

```
#tfidf of project_title
vec_tfidf_title = TfidfVectorizer(min_df=10,max_features=5000)
vec_tfidf_title.fit(x_train['project_title'])

title_tfidf_train = vec_tfidf_title.transform(x_train['project_title'])
title_tfidf_test = vec_tfidf_title.transform(x_test['project_title'])

print("test ",title_tfidf_test.shape)
print("train ",title_tfidf_train.shape)
```

```
test  (16500, 1562)
train  (33500, 1562)
```

3.3 categorical data- project_subject_categories(one hot encoding)

```
vec_projcat = CountVectorizer()
vec_projcat.fit(x_train['project_subject_categories'].values)
```

```
projcat_train = vec_projcat.transform(x_train['project_subject_categories'].values)
projcat_test = vec_projcat.transform(x_test['project_subject_categories'].values)


print(vec_projcat.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",projcat_train.shape)
print("Shape of matrix of Test data after one hot encoding ",projcat_test.shape)
```

```
['appliedlearning', 'appliedlearning_health_sports', 'appliedlearning_history_civics', 'applied
Shape of matrix of Train data after one hot encoding  (33500, 49)
Shape of matrix of Test data after one hot encoding  (16500, 49)
```

3.4 categorical data- project_subject_subcategories(one hot encoding)

```
vec_projsubcat = CountVectorizer()
vec_projsubcat.fit(x_train['project_subject_subcategories'].values)

projsubcat_train = vec_projsubcat.transform(x_train['project_subject_subcategories'].values)
projsubcat_test = vec_projsubcat.transform(x_test['project_subject_subcategories'].values)


print(vec_projsubcat.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",projsubcat_train.shape)
print("Shape of matrix of Test data after one hot encoding ",projsubcat_test.shape)
```

```
['appliedsciences', 'appliedsciences_charactereducation', 'appliedsciences_civics_government',
Shape of matrix of Train data after one hot encoding  (33500, 368)
Shape of matrix of Test data after one hot encoding  (16500, 368)
```

3.5 categorical data- school_state(one hot encoding)

```
vec_state = CountVectorizer()
vec_state.fit(x_train['school_state'].values)

state_train = vec_state.transform(x_train['school_state'].values)
state_test = vec_state.transform(x_test['school_state'].values)


print(vec_state.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",state_train.shape)
print("Shape of matrix of Test data after one hot encoding ",state_test.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in
Shape of matrix of Train data after one hot encoding  (33500, 51)
Shape of matrix of Test data after one hot encoding  (16500, 51)
```

3.6 categorical data- project_grade_category(one hot encoding)

```
vec_grade = CountVectorizer()
vec_grade.fit(x_train['project_grade_category'].values)

grade_train = vec_grade.transform(x_train['project_grade_category'].values)
grade_test = vec_grade.transform(x_test['project_grade_category'].values)

print(vec_grade.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",grade_train.shape)
print("Shape of matrix of Test data after one hot encoding ",grade_test.shape)
```

```
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
Shape of matrix of Train data after one hot encoding  (33500, 4)
Shape of matrix of Test data after one hot encoding  (16500, 4)
```

3.7 categorical data- teacher_prefix(one hot encoding)

```
vec_prefix = CountVectorizer()
vec_prefix.fit(x_train['teacher_prefix'].values)

prefix_train = vec_prefix.transform(x_train['teacher_prefix'].values)
prefix_test = vec_prefix.transform(x_test['teacher_prefix'].values)


print(vec_prefix.get_feature_names())

print("Shape of matrix of Train data after one hot encoding ",prefix_train.shape)
print("Shape of matrix of Test data after one hot encoding ",prefix_test.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix of Train data after one hot encoding  (33500, 5)
Shape of matrix of Test data after one hot encoding  (16500, 5)
```

3.8 Numerical features- total_cost(normalization)

```
normalizer = Normalizer()

# normalizer.fit(x_train['total_cost'].values)
# 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.

normalizer.fit(x_train['total_cost'].values.reshape(-1, 1))

price_train = normalizer.transform(x_train['total_cost'].values.reshape(-1, 1))
price_test = normalizer.transform(x_test['total_cost'].values.reshape(-1, 1))

#price_train=price_train.T
#price_test=price_test.T

print("After vectorizations")
print(price_train.shape, y_train.shape)
print(price_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
==================================================================================================
```

3.9 Numerical features- teacher_number_of_previously_posted_projects(normalization)

```
normalizer = Normalizer()

# normalizer.fit(x_train['total_cost'].values)
# 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.
# https://imgur.com/ldZA1zg
normalizer.fit(x_train['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))

teacher_number_of_previously_posted_projects_train = normalizer.transform(x_train['teacher_number_of
teacher_number_of_previously_posted_projects_test = normalizer.transform(x_test['teacher_number_of_p

print("After vectorizations")
print(teacher_number_of_previously_posted_projects_train.shape, y_train.shape)
print(teacher_number_of_previously_posted_projects_test.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
==================================================================================================
```

4.1 NB on feature Set 1: categorical, numerical features + preprocessed_eassay (BOW)

```
X_tr = hstack((essay_bow_train,title_bow_train,projcat_train,projsubcat_train,state_train,grade_trai
X_te = hstack((essay_bow_test,title_bow_test,projcat_test,projsubcat_test,state_test,grade_test,pref


print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(33500, 7041) (33500,)
(16500, 7041) (16500,)
==================================================================================================
```

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positi
```

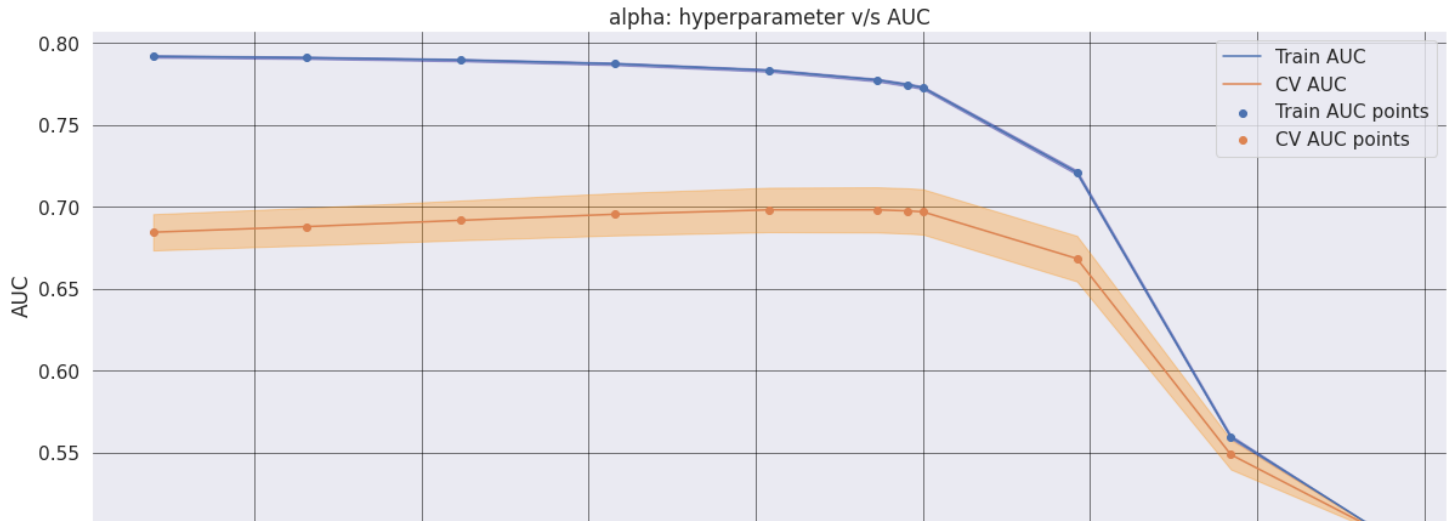
```
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred


#Gridsearch-cv with cv = 10
nb = MultinomialNB(class_prior=[0.5,0.5])
parameters = {'alpha':[0.00001, 0.0001,0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]}
clf = GridSearchCV(nb, parameters, cv= 10, scoring='roc_auc',return_train_score=True,verbose=2)
clf.fit(X_tr, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

```
[CV] END ..........................................alpha=0.1; total time=   0.1s
[CV] END ..........................................alpha=0.1; total time=   0.1s
[CV] END ..........................................alpha=0.1; total time=   0.1s
[CV] END ..........................................alpha=0.1; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.5; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ..........................................alpha=0.8; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ............................................alpha=1; total time=   0.1s
[CV] END ...........................................alpha=10; total time=   0.1s
[CV] END ...........................................alpha=10; total time=   0.1s
[CV] END ...........................................alpha=10; total time=   0.1s
[CV] END ...........................................alpha=10; total time=   0.1s
```

```
[CV] END ....................................alpha=10; total time=   0.1s
[CV] END ....................................alpha=10; total time=   0.1s
[CV] END ....................................alpha=10; total time=   0.1s
[CV] END ....................................alpha=10; total time=   0.1s
[CV] END ....................................alpha=10; total time=   0.1s
[CV] END ....................................alpha=10; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s

[CV] END ..................................alpha=1000; total time=   0.1s
[CV] END ..................................alpha=1000; total time=   0.1s
[CV] END ..................................alpha=1000; total time=   0.1s
[CV] END ..................................alpha=1000; total time=   0.1s
```

```
alphas = [0.00001, 0.0001, 0.001, 0.01, 0.1,0.5,0.8, 1, 10,  100,  1000]
log_alphas =[]

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)

plt.figure(figsize=(20,8))

plt.plot(log_alphas, train_auc, label='Train AUC')
# https://stackoverflow.com/a/48803361/4084039

plt.gca().fill_between(log_alphas,train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.3,colo
plt.plot(log_alphas, cv_auc, label='CV AUC')
plt.gca().fill_between(log_alphas,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorang
plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

```
100%|██████████| 11/11 [00:00<00:00, 20432.84it/s]
```

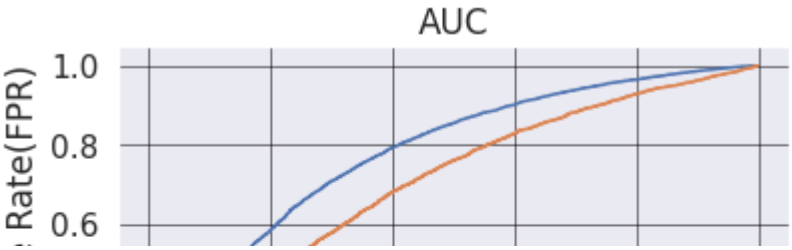


- **As both the curves cross 0.0 , we can observe a steep decline in AUC score and they converge rapidly**

```
best_alpha1=clf.best_params_
print(best_alpha1)
```

```
{'alpha': 0.5}
```

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.r
from sklearn.metrics import roc_curve, auc

nb_bow = MultinomialNB(alpha = 0.5,class_prior=[0.5,0.5])

nb_bow.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive c
# not the predicted outputs

y_train_pred = batch_predict(nb_bow, X_tr)
y_test_pred = batch_predict(nb_bow, X_te)

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("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



Summary

- **For Bow model for alpha=0.5 ,we get train AUC of 0.77 and Test AUC of 0.68**

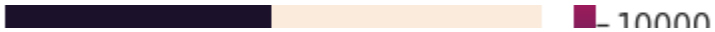
Confusion matrix train data

```
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(fpr*(1-tpr))]

    # (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)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions


print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))

conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, t

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999996255834908 for threshold 0.034
[[ 2583  2585]
 [ 4111 24221]]
the maximum value of tpr*(1-fpr) 0.24999996255834908 for threshold 0.034
```

Summary

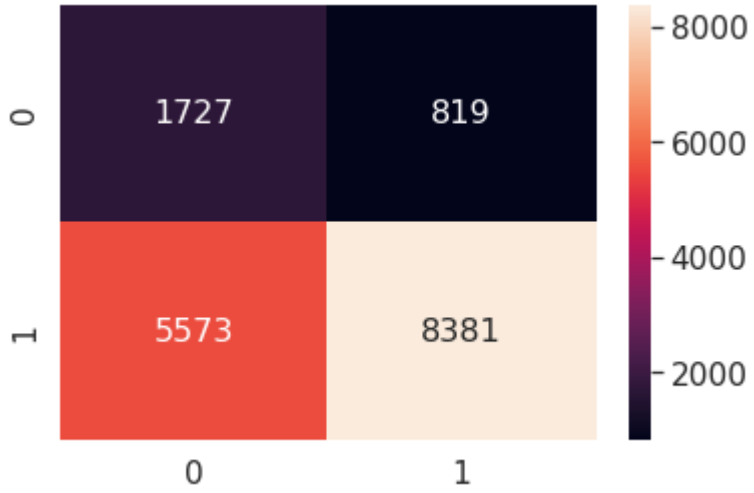
- In the following confusion matrix we observe that the model has **24221 true positives** while **2583 true negatives**.
- False negatives are roughly close to **4000** which indeed is large.



Confusion matrix test data

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))

conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.779
[[1727  819]
 [5573 8381]]
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.779
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c5493ffd0>
```



Summary

- There are **8381 true positives**
- This also has large number of false negatives: **5573**.

4.2 NB on feature Set 2: categorical, numerical features + preprocessed_eassay (TFIDF)

```
X_tr = hstack((essay_tfidf_train,title_tfidf_train,projcat_train,projsubcat_train,state_train,grade_
X_te = hstack((essay_tfidf_test,title_tfidf_test,projcat_test,projsubcat_test,state_test,grade_test,


print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(33500, 7041) (33500,)
(16500, 7041) (16500,)
====================================================================================================
```

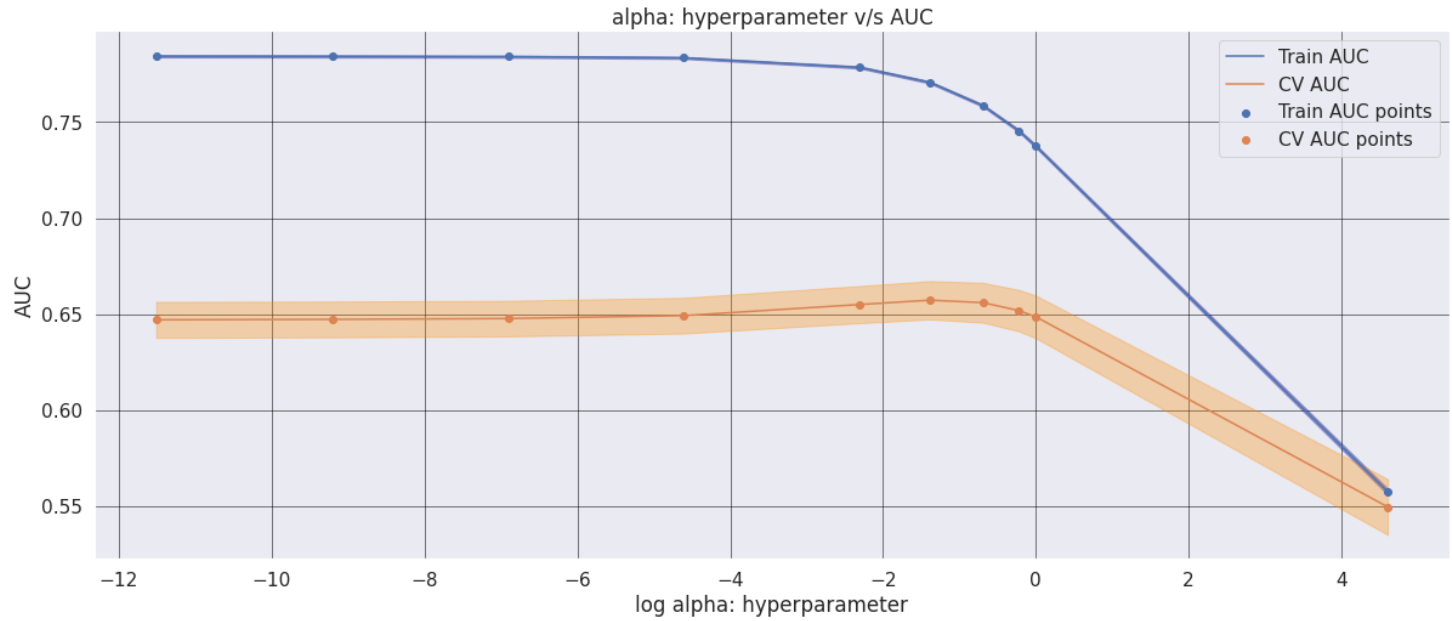
```
#Gridsearch-cv with cv = 10
nb = MultinomialNB(class_prior=[0.5,0.5])

parameters = {'alpha':[0.00001, 0.0001, 0.001, 0.01, 0.1,0.25,0.5,0.8, 1,100]}

clf = GridSearchCV(nb, parameters, cv= 10, scoring='roc_auc',return_train_score=True,verbose=2)

clf.fit(X_tr, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

```
[CV] END .....................................alpha=0.01; total time=   0.1s
[CV] END .....................................alpha=0.01; total time=   0.1s
[CV] END .....................................alpha=0.01; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END ......................................alpha=0.1; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END .....................................alpha=0.25; total time=   0.1s
[CV] END ......................................alpha=0.5; total time=   0.1s
[CV] END ......................................alpha=0.5; total time=   0.1s
[CV] END ......................................alpha=0.5; total time=   0.1s
```

```
[CV] END .....................................alpha=0.5; total time=   0.1s
[CV] END .....................................alpha=0.5; total time=   0.1s
[CV] END .....................................alpha=0.5; total time=   0.1s
[CV] END .....................................alpha=0.5; total time=   0.1s
[CV] END .....................................alpha=0.5; total time=   0.1s
[CV] END .....................................alpha=0.5; total time=   0.1s
[CV] END .....................................alpha=0.5; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=0.8; total time=   0.1s
[CV] END .....................................alpha=1; total time=   0.1s
[CV] END .....................................alpha=1; total time=   0.1s
[CV] END .....................................alpha=1; total time=   0.1s
[CV] END .....................................alpha=1; total time=   0.1s
[CV] END .....................................alpha=1; total time=   0.1s
[CV] END .....................................alpha=1; total time=   0.1s
[CV] END .....................................alpha=1; total time=   0.1s
[CV] END .....................................alpha=1; total time=   0.1s

[CV] END .....................................alpha=1; total time=   0.1s
[CV] END .....................................alpha=1; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
[CV] END ...................................alpha=100; total time=   0.1s
```

```
alphas = [0.00001, 0.0001, 0.001, 0.01, 0.1,0.25,0.5,0.8, 1,100]
log_alphas =[]

for a in tqdm(alphas):
    b = math.log(a)
    log_alphas.append(b)

plt.figure(figsize=(20,8))

plt.plot(log_alphas, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alphas,train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.3,colo

plt.plot(log_alphas, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(log_alphas,cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.3,color='darkorang

plt.scatter(log_alphas, train_auc, label='Train AUC points')
plt.scatter(log_alphas, cv_auc, label='CV AUC points')


plt.legend()
plt.xlabel("log alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```

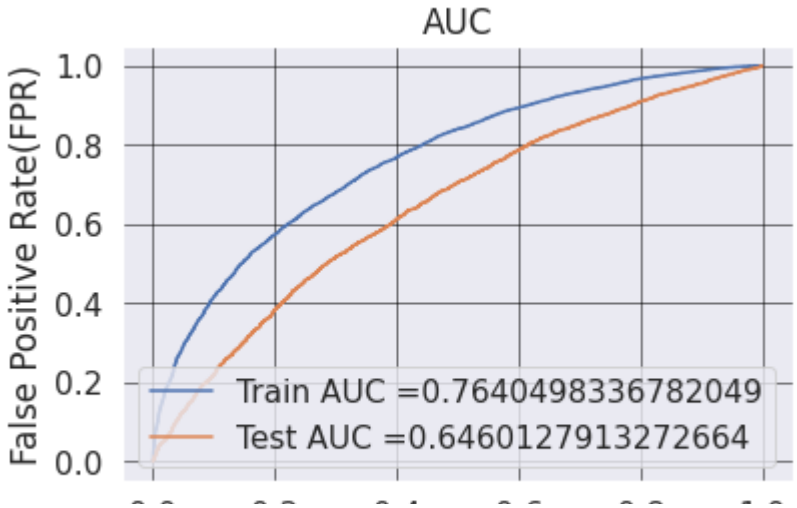
```
100%|██████████| 10/10 [00:00<00:00, 61410.01it/s]
```




```
best_alpha2=clf.best_params_
print(best_alpha2)
```

```
{'alpha': 0.25}
```

```
nb_tfidf = MultinomialNB(alpha = 0.25,class_prior=[0.5,0.5])

nb_tfidf.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive c
# not the predicted outputs

y_train_pred = batch_predict(nb_tfidf, X_tr)
y_test_pred = batch_predict(nb_tfidf, X_te)

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("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



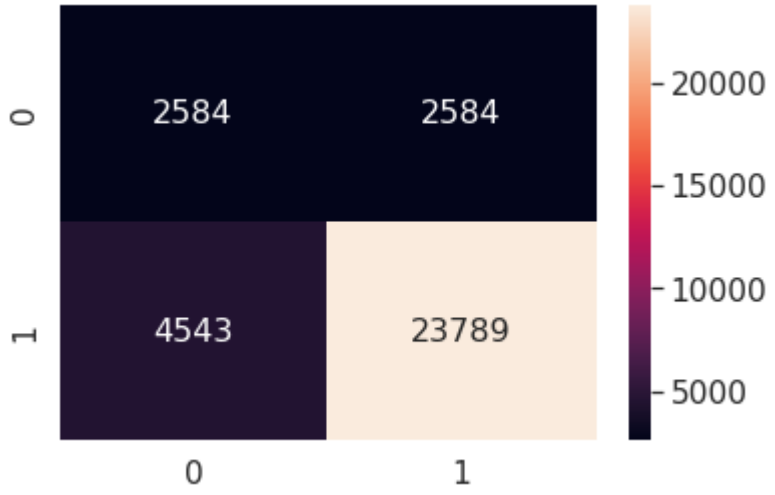
Summary

- For **tfidf** model for **alpha=0.05** ,we get **train AUC of 0.76** and **Test AUC of 0.64**.

Confusion matrix train data

```
print("="*100)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr)))

conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, t

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.359
[[ 2584  2584]
 [ 4543 23789]]
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.359
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c67e67e10>
```



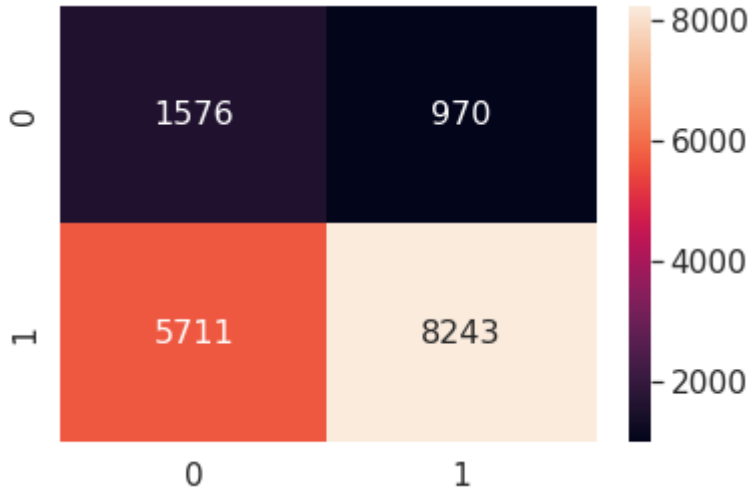
Summary

- In the following confusion matrix we observe that the model has **23789 true positives** while **2584 true negatives**.
- It has large number of false negatives which are roughly close to **4500**.

Confusion matrix test data

```
print("="*100)
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))

conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test

sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================================
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.556
[[1576  970]
 [5711 8243]]
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.556
<matplotlib.axes._subplots.AxesSubplot at 0x7f8c54cb9bd0>
```



Summary

- There are **8243 true positives**
- This also has large number of false negatives: **5711**.

5. top 20 features(names not the indexes)

Using set2

```
# https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes/50530697
features=[]

for temp in vec_projcat.get_feature_names() :
```

```
    features.append(temp)
for temp in vec_projsubcat.get_feature_names() :
    features.append(temp)
for temp in vec_state.get_feature_names() :
    features.append(temp)
for temp in vec_grade.get_feature_names() :
    features.append(temp)
for temp in vec_prefix.get_feature_names() :
    features.append(temp)
for temp in vec_tfidf_title.get_feature_names() :
    features.append(temp)
for temp in vec_tfidf_essay.get_feature_names() :
    features.append(temp)
features.append("price")
features.append("teacher_number_of_previously_posted_projects")


print("Positive features:")
sorted_tfidf_pos = nb_tfidf.feature_log_prob_[1, :].argsort()[::-1][:len(features)]
for i in sorted_tfidf_pos[0:20]:
    print(features[i])
```

```
 Positive features:
 price
 teacher_number_of_previously_posted_projects
 zone
 zero
 zones
 youngest
 thought
 youth
 thriving
 workout
 imovie
 thousands
 youtube
 zip
 therapy
 wearing
 web
 wrote
 years
 workshop
```

```
print("negative features:")
sorted_tfidf_neg = nb_tfidf.feature_log_prob_[0, :].argsort()[::-1][:len(features)]
for i in sorted_tfidf_neg[0:20]:
    print(features[i])
```

```
 negative features:
 price
 teacher_number_of_previously_posted_projects
 zone
 zero
 zones
 youngest
 thought
 thriving
 youth
 imovie
 workout
```

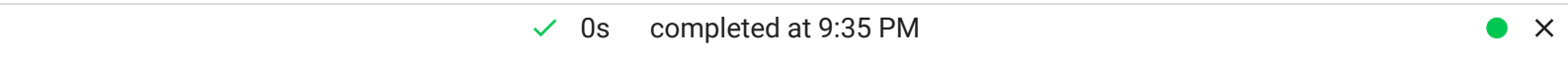
```
thousands
youtube
zip
therapy
years
web
workshop
wearing
wrote
```

6. Conclusion

```
# https://stackoverflow.com/questions/36423259/how-to-use-pretty-table-in-python-to-print-out-data-f

from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Alpha: Hyper-parameter", " Test AUC"]

x.add_row(["BOW", "Naive Bayes", 0.5, 0.68])
x.add_row(["TFIDF", "Naive Bayes", 0.25, 0.64])

print(x)
```

```
+------------+-------------+------------------------+-----------+
| Vectorizer |    Model    | Alpha: Hyper-parameter |  Test AUC |
+------------+-------------+------------------------+-----------+
|    BOW     | Naive Bayes |          0.5           |    0.68   |
|   TFIDF    | Naive Bayes |          0.25          |    0.64   |
+------------+-------------+------------------------+-----------+
```

- There is not much difference between the test-AUC score of set 1 features and set 2 features.
- Comparatively better than KNN.
- **Why is it important to choose the best hyperparameter(alpha) vlaue:** Basically we want to decrease the effect of rare words: for example if you have one spam email with the word 'money' in it, and no nonspam emails with this word, then without additive smoothing, your spam filter will classify every email with this keyword as spam. More about additive smoothing refer here on wikipedia.
- **Why did we use log scale for alpha on x_axis:** Log naturally reduces the dynamic range of a variable so the differences are preserved while the scale is not that dramatically skewed. **Example:** Imagine some people got 100,000,000 loan and some got 10000 and some 0. Any feature scaling is probably gonna put 0 and 10000 so close to each other as the biggest number anyway pushes the boundary. Logarithm solves the issue.
- **cv=10 in GridSearchCV() function:** when using cross validation with cv=10 the data is split into 10 parts i.e. 10% / 90%, then each part is used for training while the rest used for validation.


0s
completed at 9:35 PM