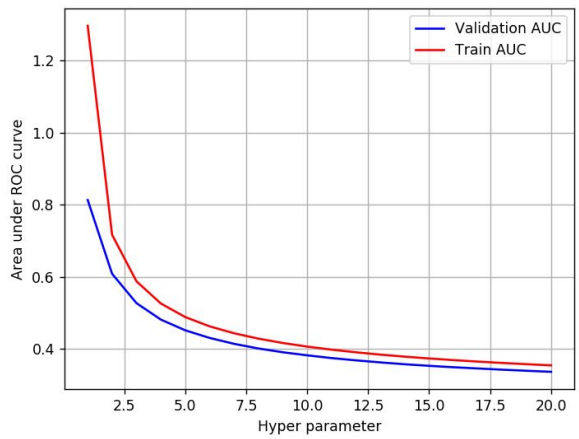
Assignment 6: Apply NB

1. Minimum data points need to be considered for people having 4GB RAM is **50k** and for 8GB RAM is **100k**
2. When you are using ramdomsearchcv or gridsearchcv you need not split the data into X_train,X_cv,X_test. As the above methods use kfold. The model will learn better if train data is more so splitting to X_train,X_test will suffice.
3. If you are writing for loops to tune your model then you need split the data into X_train,X_cv,X_test.
4. While splitting the data explore stratify parameter.
5. **Apply Multinomial NB on these feature sets**
   - Features that need to be considered

     essay
     while encoding essay, try to experiment with the max_features and n_grams parameter of vectorizers and see if it increases AUC score.
     categorical features
     - teacher_prefix
     - project_grade_category
     - school_state
     - clean_categories
     - clean_subcategories
     numerical features
     - price
     - teacher_number_of_previously_posted_projects
     while encoding the numerical features check this and this

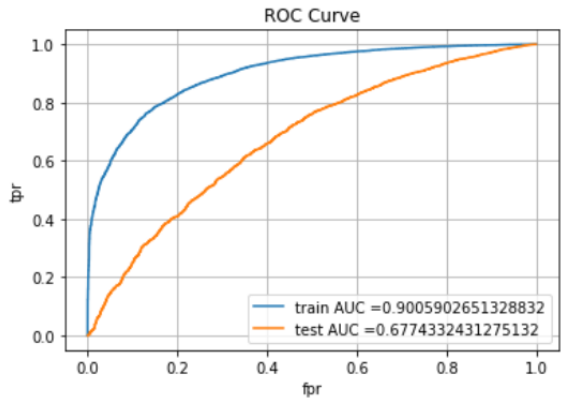
   - Set 1: categorical, numerical features + preprocessed_eassay (BOW)
   - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF)
6. **The hyper paramter tuning(find best alpha:smoothing parameter)**
   - Consider alpha values in range: 10^-5 to 10^2 like [0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]
   - Explore class_prior = [0.5, 0.5] parameter which can be present in MultinomialNB function(go through this) then check how results might change.
   - Find the best hyper parameter which will give the maximum AUC value
   - For hyper parameter tuning using k-fold cross validation(use GridsearchCV or RandomsearchCV)/simple cross validation data (write for loop to iterate over hyper parameter values)

- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



-while plotting take log(alpha) on your X-axis so that it will be more readable

- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.



- Along with plotting ROC curve, you need to print the [confusion matrix]() with predicted

	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

and original labels of test data points

-plot the confusion matrix in heatmaps, while plotting the confusion matrix go through the [link]()

7. find the top 20 features from either from feature Set 1 or feature Set 2 using values of `feature_log_prob_ ` parameter of `MultinomialNB` (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) and print **BOTH** positive as well as negative corresponding feature names.

   - go through the [link]()

8. You need to summarize the results at the end of the notebook, summarize it in the table format

Vectorizer	Model	Hyper parameter	AUC
BOW	Brute	7	0.78
TFIDF	Brute	12	0.79
W2V	Brute	10	0.78
TFIDFW2V	Brute	6	0.78

2. Naive Bayes

1.1 Loading Data

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
import matplotlib.pyplot as plt
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
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import roc_curve, auc
import seaborn as sns; sns.set()
import re
from nltk.corpus import stopwords
import pickle
from tqdm import tqdm
import os
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

```
from google.colab import drive
drive.mount('drive')
```

```
Mounted at drive
```

```
train_path="/content/drive/My Drive/train_data.csv"
resource_path="/content/drive/My Drive/resources.csv"
```

```
import pandas
project_data = pd.read_csv(train_path,nrows=50000)
resource_data=pd.read_csv(resource_path)
```

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_sta
 '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']
```

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

	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

preprocessing categorical and numerical data

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

```
Grades PreK-2    20316
Grades 3-5       16968
Grades 6-8        7750
Grades 9-12       4966
Name: project_grade_category, dtype: int64
```

```
#Preprocessing project grade category
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-s
project_data['project_grade_category'] = project_data['project_grade_category'].str.replac
project_data['project_grade_category'] = project_data['project_grade_category'].str.replac
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower(
```

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

```
grades_prek_2    20316
grades_3_5       16968
grades_6_8        7750
grades_9_12       4966
Name: project_grade_category, dtype: int64
```

```
project_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
Health & Sports, History & Civics                25
History & Civics, Applied Learning               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
```

```
#Preprocessing project_subject_categories
project_data['project_subject_categories'] = project_data['project_subject_categories'].st
project_data['project_subject_categories'] = project_data['project_subject_categories'].st
project_data['project_subject_categories'] = project_data['project_subject_categories'].st
project_data['project_subject_categories'] = project_data['project_subject_categories'].st
project_data['project_subject_categories'] = project_data['project_subject_categories'].st
project_data['project_subject_categories'].value_counts()
```

```
literacy_language                       10927
math_science                             7695
literacy_language_math_science           6705
health_sports                            4700
music_arts                               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_arts              794
math_science_music_arts                   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_arts                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_arts                   140
history_civics_music_arts                 135
health_sports_math_science                118
history_civics_specialneeds               103
health_sports_appliedlearning              99
appliedlearning_history_civics             78
music_arts_specialneeds                    67
health_sports_music_arts                   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_arts_health_sports                   10
music_arts_history_civics                   9
appliedlearning_warmth_care_hunger          8
history_civics_health_sports                8
math_science_warmth_care_hunger             7
specialneeds_warmth_care_hunger             6
music_arts_appliedlearning                  4
literacy_language_warmth_care_hunger        3
```

```
music_arts_warmth_care_hunger               1
Name: project_subject_categories, dtype: int64
```

```
project_data['teacher_prefix'].value_counts()
```

```
Mrs.       26140
Ms.        17936
Mr.         4859
Teacher     1061
Dr.            2
Name: teacher_prefix, dtype: int64
```

```
#Preprocessing teacher_prefix
# check if we have any nan values are there
print(project_data['teacher_prefix'].isnull().values.any())
print("number of nan values",project_data['teacher_prefix'].isnull().values.sum())
```

```
True
number of nan values 2
```

```
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
project_data['teacher_prefix'].value_counts()
```

```
Mrs.       26142
Ms.        17936
Mr.         4859
Teacher     1061
Dr.            2
Name: teacher_prefix, dtype: int64
```

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

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

```
project_data['project_subject_subcategories'].value_counts()
```

```
Literacy                                    4434
Literacy, Mathematics                       3833
Literature & Writing, Mathematics           2705
Literacy, Literature & Writing              2570
Mathematics                                 2441
                                            ...
Parent Involvement, Team Sports                1
Financial Literacy, Visual Arts                1
Community Service, Gym & Fitness               1
Character Education, Financial Literacy        1
Financial Literacy, Social Sciences            1
Name: project_subject_subcategories, Length: 384, dtype: int64
```

```
#Preprocessing project_subject_subcategories
project_data['project_subject_subcategories'] = project_data['project_subject_subcategorie
project_data['project_subject_subcategories'] = project_data['project_subject_subcategorie
project_data['project_subject_subcategories'] = project_data['project_subject_subcategorie
project_data['project_subject_subcategories'] = project_data['project_subject_subcategorie
project_data['project_subject_subcategories'] = project_data['project_subject_subcategorie
project_data['project_subject_subcategories'].value_counts()
```

```
literacy                                   4434
literacy_mathematics                       3833
literature_writing_mathematics             2705
literacy_literature_writing                2570
mathematics                                2441
                                           ...
gym_fitness_parentinvolvement                 1
college_careerprep_warmth_care_hunger         1
charactereducation_economics                  1
environmentalscience_financialliteracy        1
college_careerprep_gym_fitness                1
Name: project_subject_subcategories, Length: 384, dtype: int64
```

```
project_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
```

```
#preprocessing school_state
project_data['school_state'] = project_data['school_state'].str.lower()
project_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
tn     774
ct     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
```

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

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

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

```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'hi
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', '
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', '
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over'
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', '
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'd
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn'
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn
            'won', "won't", 'wouldn', "wouldn't"]
```

```
project_data['project_title'].head(5)
```

```
0     Educational Support for English Learners at Home
```

```
1                Wanted: Projector for Hungry Learners
2    Soccer Equipment for AWESOME Middle School Stu...
3                              Techie Kindergarteners
4                              Interactive Math Tools
Name: project_title, dtype: object
```

```
print("printing some random reviews")
print(9, project_data['project_title'].values[9])
print(34, project_data['project_title'].values[34])
print(147, project_data['project_title'].values[147])
```

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

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

```
#preprocessing project_title
preprocessed_titles = preprocess_text(project_data['project_title'].values)
```

```
100%|██████████| 50000/50000 [00:01<00:00, 42347.28it/s]
```

```
print("printing some random reviews")
print(9, preprocessed_titles[9])
print(34, preprocessed_titles[34])
print(147, preprocessed_titles[147])
```

```
printing some random reviews
9 love reading pure pleasure
34 ball
147 needs chromebook
```

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

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

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

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


sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and language de
==================================================
```

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

```
My kindergarten students have varied disabilities ranging from speech and language de
```

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language de
```

```
# 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'hi
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', '
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', '
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over'
```

```
 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any',
 'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', '
 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'd
 "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn'
 "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn
 'won', "won't", 'wouldn', "wouldn't"]
```

```
#convert all the words to lower case first and then remove the stopwords
for i in range(len(project_data['essay'].values)):
    project_data['essay'].values[i] = project_data['essay'].values[i].lower()
```

```
# 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 = sent.replace('nan',' ')
    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_essays.append(sent.lower().strip())
```

```
100%|██████████| 50000/50000 [00:27<00:00, 1813.74it/s]
```

```
#creating a new column with the preprocessed essays and replacing it with the original col
project_data['preprocessed_essays'] = preprocessed_essays
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)
```

```
#convert all the words to lower case first and then remove the stopwords
for i in range(len(project_data['project_title'].values)):
    project_data['project_title'].values[i] = project_data['project_title'].values[i].lowe
```

```
# similarly you can preprocess the titles also
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 = sent.replace('nan',' ')
    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)
```

```
    sent = .join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 50000/50000 [00:01<00:00, 42349.31it/s]
```

```
#creating a new column with the preprocessed titles,useful for analysis
project_data['preprocessed_titles'] = preprocessed_titles
```

```
y = project_data['project_is_approved'].values
X = project_data.drop(['project_is_approved'], axis=1)
X.head(1)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs	in

1.2 Splitting data into Train and cross validation(or test): Stratified Sampling

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
print("Split ratio")
print('-'*50)
print('Train dataset:',len(X_train)/len(X)*100,'%\n','size:',len(X_train))
print('Test dataset:',len(X_test)/len(X)*100,'%\n','size:',len(X_test))
```

```
Split ratio
--------------------------------------------------
Train dataset: 67.0 %
 size: 33500
Test dataset: 33.0 %
 size: 16500
```

1.3 Make Data Model Ready: encoding eassay, and project_title

```
#Converting essay in to BOW representation using count vectorizer.
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(X_train['preprocessed_essays'].values) # fit has to happen only o

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer_bow_essay.transform(X_train['preprocessed_essays'].values)
```

```
X_test_essay_bow = vectorizer_bow_essay.transform(X_test['preprocessed_essays'].values)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)

print(X_test_essay_bow.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(33500, 10335) (33500,)
(16500, 10335) (16500,)
====================================================================================
```

```
#Converting essay in to TFIDF representation using count vectorizer.
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(X_train['preprocessed_essays'])      #Fitting has to be on Trai

X_train_essay_tfidf = vectorizer_tfidf_essay.transform(X_train['preprocessed_essays'].valu

X_test_essay_tfidf = vectorizer_tfidf_essay.transform(X_test['preprocessed_essays'].values

print("Shape of train data matrix after one hot encoding ",X_train_essay_tfidf.shape)

print("Shape of test data matrix after one hot encoding ",X_test_essay_tfidf.shape)
```

```
Shape of train data matrix after one hot encoding  (33500, 10335)
Shape of test data matrix after one hot encoding  (16500, 10335)
```

1.4 Make Data Model Ready: encoding numerical, categorical features

```
#one hot encoding school_state
vectorizer1 = CountVectorizer(binary=True)
vectorizer1.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 = vectorizer1.transform(X_train['school_state'].values)
X_test_state_ohe = vectorizer1.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer1.get_feature_names())
print("="*100)
```

```
After vectorizations
(33500, 51) (33500,)
(16500, 51) (16500,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id',
====================================================================================
```

```
#one hot encoding teacher_prefix
vectorizer2 = CountVectorizer()
vectorizer2.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer2.transform(X_train['teacher_prefix'].values)

X_test_teacher_ohe = vectorizer2.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)

print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer2.get_feature_names())
print("="*100)
```

```
After vectorizations
(33500, 5) (33500,)
(16500, 5) (16500,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
====================================================================================
```

```
#One hot coding project_grade_category
vectorizer3 = CountVectorizer()
vectorizer3.fit(X_train['project_grade_category'].values) # fit has to happen only on trai

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer3.transform(X_train['project_grade_category'].values)

X_test_grade_ohe = vectorizer3.transform(X_test['project_grade_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)

print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer3.get_feature_names())
print("="*100)
```

```
After vectorizations
(33500, 4) (33500,)
(16500, 4) (16500,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
====================================================================================
```

```
#One hot encoding project_subject_categories
vectorizer4 = CountVectorizer()
vectorizer4.fit(X_train['project_subject_categories'].values) # fit has to happen only on

# we use the fitted CountVectorizer to convert the text to vector
X_train_subject_ohe = vectorizer4.transform(X_train['project_subject_categories'].values)

X_test_subject_ohe = vectorizer4.transform(X_test['project_subject_categories'].values)
```

```
print("After vectorizations")
print(X_train_subject_ohe.shape, y_train.shape)

print(X_test_subject_ohe.shape, y_test.shape)
print(vectorizer4.get_feature_names())
print("="*100)
```

```
After vectorizations
(33500, 50) (33500,)
(16500, 50) (16500,)
['appliedlearning', 'appliedlearning_health_sports', 'appliedlearning_history_civics
==========================================================================================
```

```
#One hot encoding project_subject_subcategories
vectorizer5 = CountVectorizer()
vectorizer5.fit(X_train['project_subject_subcategories'].values) # fit has to happen only

# we use the fitted CountVectorizer to convert the text to vector
X_train_subject_sub_ohe = vectorizer5.transform(X_train['project_subject_subcategories'].v

X_test_subject_sub_ohe = vectorizer5.transform(X_test['project_subject_subcategories'].val

print("After vectorizations")
print(X_train_subject_sub_ohe.shape, y_train.shape)

print(X_test_subject_sub_ohe.shape, y_test.shape)
print(vectorizer5.get_feature_names())
print("="*100)
```

```
After vectorizations
(33500, 368) (33500,)
(16500, 368) (16500,)
['appliedsciences', 'appliedsciences_charactereducation', 'appliedsciences_civics_gov
==========================================================================================
```

```
#Normalizing price
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-g
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_inde
price_data.head(2)
```

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

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

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
```

```
# normalizer.fit(X_train['price'].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['price'].values.reshape(-1,1))
X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(-1,1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))
print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)

print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

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

```
#Normalizing teacher_number_of_previously_posted_projects
normalizer = Normalizer()
# normalizer.fit(X_train['price'].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['teacher_number_of_previously_posted_projects'].values.reshape(-1,1

X_train_post_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_proj

X_test_post_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projec



print("After vectorizations")
print(X_train_post_norm.shape, y_train.shape)

print(X_test_post_norm.shape, y_test.shape)
print("="*100)
```

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

SET-1

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((X_train_essay_bow, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_oh
```

```
X_te = hstack((X_test_essay_bow, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe,X_

print("Final Data matrix")
print(X_tr.shape, y_train.shape)

print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(33500, 10815) (33500,)
(16500, 10815) (16500,)
==========================================================================================
```

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of
    # 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 tr_loop will be 49041 - 49041%1000 = 490
    # 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
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

Perfoming grid search for hyper parameter tuning bold text

```
mnb_bow = MultinomialNB(class_prior=[0.5, 0.5])
parameters = {'alpha':[0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]
clf = GridSearchCV(mnb_bow, parameters, cv=3, scoring='roc_auc',verbose=1,return_train_sco
clf_results=clf.fit(X_tr, y_train)
print(clf_results.best_score_)
print(clf_results.best_estimator_)
print(clf_results.best_params_)
```

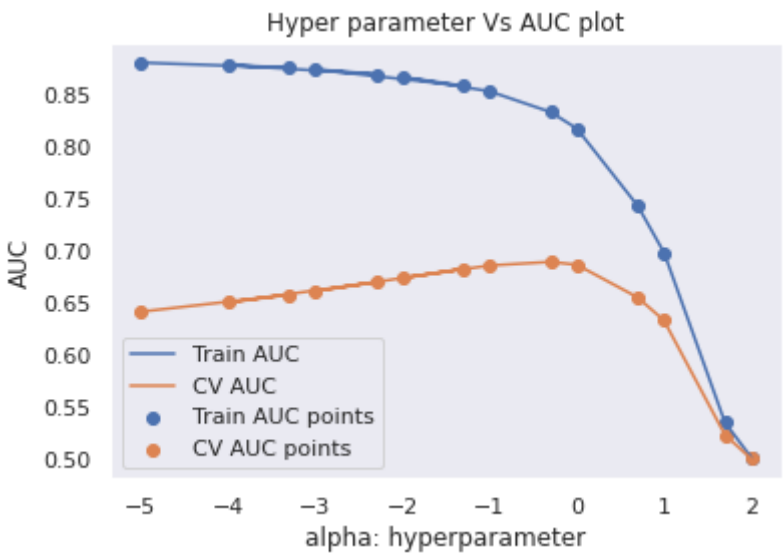
```
Fitting 3 folds for each of 14 candidates, totalling 42 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
0.6895308383128542
MultinomialNB(alpha=0.5, class_prior=[0.5, 0.5], fit_prior=True)
{'alpha': 0.5}
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:    2.6s finished
```

Plotting Hyperparameter vs AUC curve

```
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']
alpha=[0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]
log_alpha=[]
for a in tqdm(alpha):
    b = np.log10(a)
    log_alpha.append(b)


plt.plot(log_alpha, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.2,

plt.plot(log_alpha, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darko

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


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
```

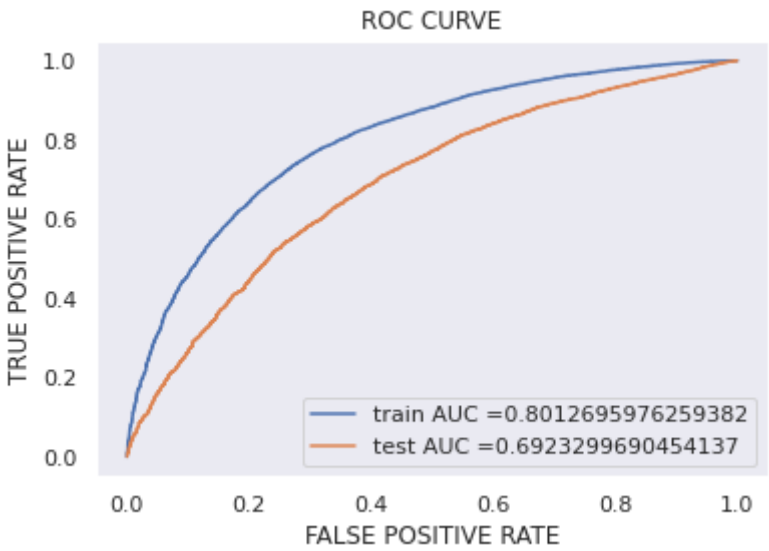
```
100%|██████████| 14/14 [00:00<00:00, 15452.70it/s]
```



1.5 Appling NB on different kind of featurization as mentioned in the instructions

```
bestalpha1=clf_results.best_params_['alpha']


# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn
```

```
from sklearn.metrics import roc_curve, auc


mnb_bow_testModel = MultinomialNB(alpha = bestalpha1,class_prior=[0.5, 0.5])
mnb_bow_testModel.fit(X_tr, y_train)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the
# not the predicted outputs

y_train_pred = mnb_bow_testModel.predict_proba(X_tr)[:,1]
y_test_pred = mnb_bow_testModel.predict_proba(X_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("ROC CURVE")
plt.grid()
plt.show()
```



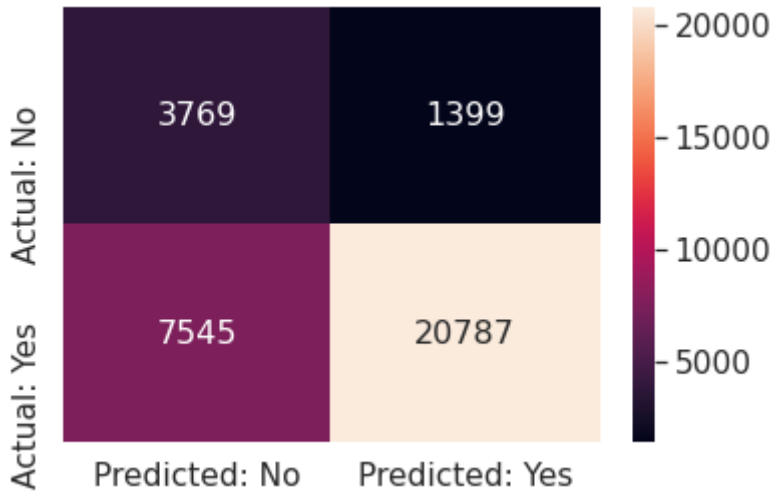

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(
    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_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
====================================================================================
the maximum value of tpr*(1-fpr) 0.5350793802607647 for threshold 0.473
Train confusion matrix
[[ 3769  1399]
 [ 7545 20787]]
Test confusion matrix
[[1499 1047]
 [4182 9772]]
```

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns; sns.set()

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_t
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
<matplotlib.axes._subplots.AxesSubplot at 0x7f2023a50320>
```



SET-2

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr1 = hstack((X_train_essay_tfidf, X_train_state_ohe, X_train_teacher_ohe, X_train_grade

X_te1 = hstack((X_test_essay_tfidf, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe
```

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)

print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(33500, 10815) (33500,)
(16500, 10815) (16500,)
====================================================================================
```

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of
    # 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 tr_loop will be 49041 - 49041%1000 = 490
    # 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
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

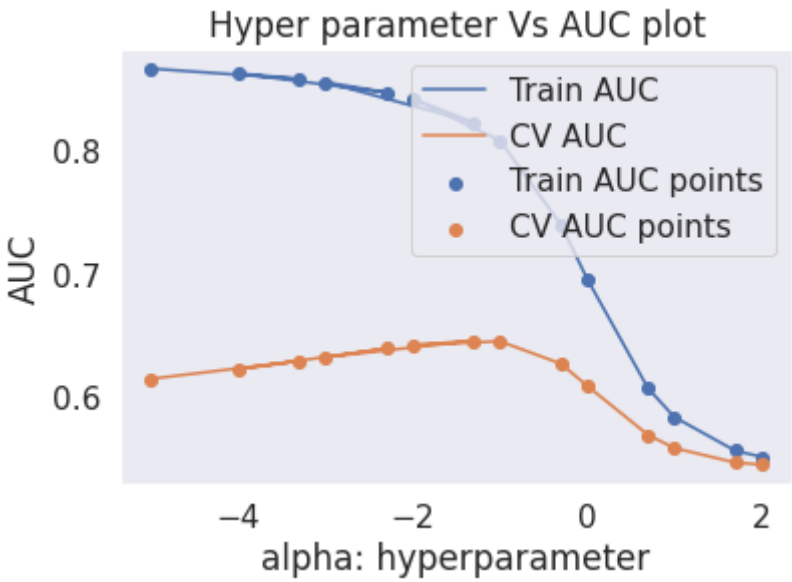
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.h
from sklearn.model_selection import GridSearchCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.neighbors import KNeighborsClassifier
from scipy.stats import randint as sp_randint
from sklearn.model_selection import RandomizedSearchCV

mnb_bow = MultinomialNB(class_prior=[0.5, 0.5])
parameters = {'alpha':[0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]
clf = GridSearchCV(mnb_bow, parameters, cv=3, scoring='roc_auc',verbose=1,return_train_sco
clf_results=clf.fit(X_tr1, y_train)
print(clf_results.best_score_)
print(clf_results.best_estimator_)
print(clf_results.best_params_)
```

```
Fitting 3 folds for each of 14 candidates, totalling 42 fits
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
0.6459531319898333
MultinomialNB(alpha=0.05, class_prior=[0.5, 0.5], fit_prior=True)
{'alpha': 0.05}
[Parallel(n_jobs=1)]: Done  42 out of  42 | elapsed:    2.6s finished
```

```
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']
```

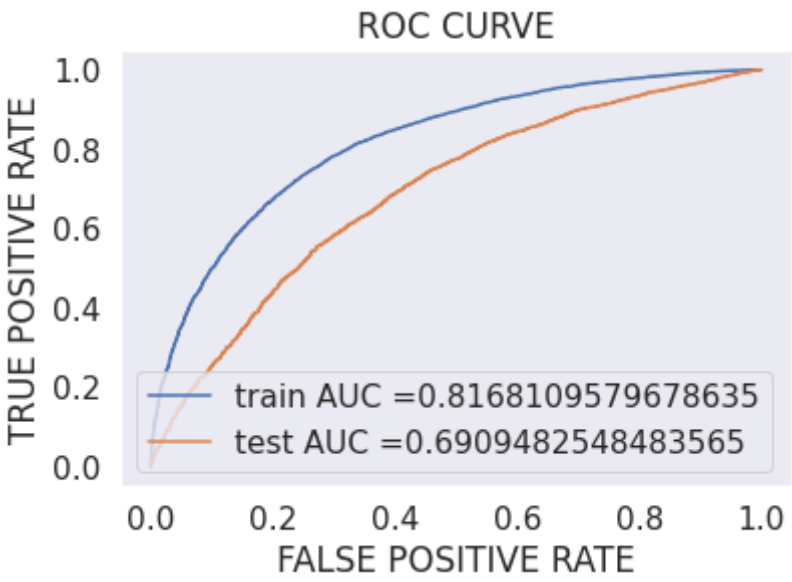
```
cv_auc_std= clf.cv_results_['std_test_score']
alpha=[0.00001,0.0005, 0.0001,0.005,0.001,0.05,0.01,0.1,0.5,1,5,10,50,100]
log_alpha=[]
for a in tqdm(alpha):
    b = np.log10(a)
    log_alpha.append(b)


plt.plot(log_alpha, train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.2,

plt.plot(log_alpha, cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# plt.gca().fill_between(K, cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darko

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


plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyper parameter Vs AUC plot")
plt.grid()
plt.show()
```

```
100%|██████████| 14/14 [00:00<00:00, 46272.86it/s]
```




```
bestalpha_2=clf_results.best_params_['alpha']


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


mnb_bow_testModel = MultinomialNB(alpha = bestalpha_2,class_prior=[0.5, 0.5])
mnb_bow_testModel.fit(X_tr, y_train)
```

```
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the
# not the predicted outputs

y_train_pred = mnb_bow_testModel.predict_proba(X_tr)[:,1]
y_test_pred = mnb_bow_testModel.predict_proba(X_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("ROC CURVE")
plt.grid()
plt.show()
```



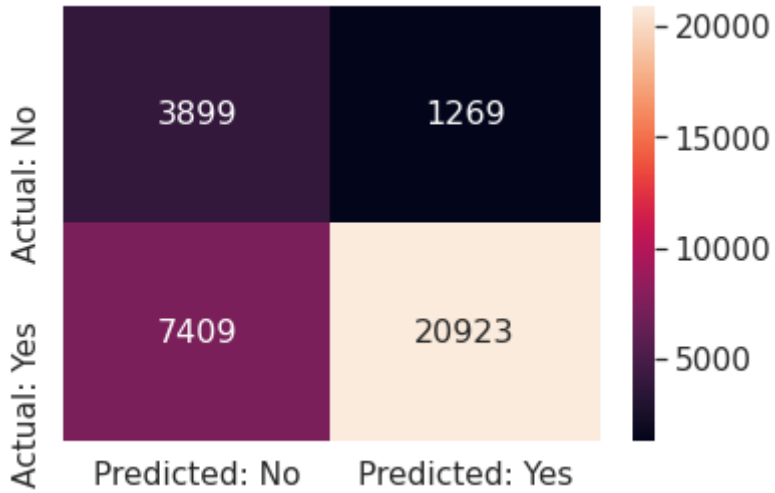

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(
    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_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
====================================================================================
the maximum value of tpr*(1-fpr) 0.5518149064782069 for threshold 0.554
Train confusion matrix
[[ 3891  1277]
 [ 7567 20765]]
Test confusion matrix
[[1502 1044]
 [4232 9722]]
```

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns; sns.set()

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_train, predict_with_best_t(y_t
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
Train data confusion matrix
<matplotlib.axes._subplots.AxesSubplot at 0x7f0c58def860>
```



Top 20 features in positive and negative

```
#this code taken from https://imgur.com/mWvE7gj
all_feature_names_bow=[]
for i in vectorizer1.get_feature_names():
    all_feature_names_bow.append(i)

for i in vectorizer2.get_feature_names():
    all_feature_names_bow.append(i)

for i in vectorizer3.get_feature_names():
```

```
    all_feature_names_bow.append(i)

for i in vectorizer4.get_feature_names():
    all_feature_names_bow.append(i)

for i in vectorizer5.get_feature_names():
    all_feature_names_bow.append(i)


for i in vectorizer_bow_essay.get_feature_names():
    all_feature_names_bow.append(i)
all_feature_names_bow.append("price")
all_feature_names_bow.append("teacher_number_of_previously_posted_projects")

print( len(all_feature_names_bow))
```

```
10815
```

```
#for BOW
nb = MultinomialNB(alpha=0.5,class_prior=[0.5,0.5])
nb.fit(X_tr,y_train)# fit the model
# now make a dictionary of all the probabilities fo the weights
bow_features_probs = []
for a in range(10815):
    bow_features_probs.append(nb.feature_log_prob_[0,a] )
print(len(bow_features_probs))
```

```
10815
```

```
#top 20 negatives
final_bow_features = pd.DataFrame({'feature_prob_estimates' : bow_features_probs, 'feature
a =final_bow_features.sort_values(by = ['feature_prob_estimates'], ascending = False)
a.head(20)
```

	feature_prob_estimates	feature_names
8936	-2.986846	sleeping
8107	-4.092037	related
5368	-4.391725	insights
1736	-4.559083	broadcast
6246	-4.748441	meantime
5364	-4.758695	insert
4420	-4.767545	friends
10813	-4.910452	price

```
#top 20 Positives
bow_features_probs_pos = []
for a in range(10815):
    bow_features_probs_pos.append(nb.feature_log_prob_[1,a] )# positive feature probabilit
#len(bow_features_probs)
final_bow_features = pd.DataFrame({'feature_prob_estimates_pos' : bow_features_probs_pos,'
a =final_bow_features.sort_values(by = ['feature_prob_estimates_pos'], ascending = False)
a.head(20)
```

	feature_prob_estimates_pos	feature_names
8936	-2.968779	sleeping
8107	-4.116734	related
1756	-4.300795	broadcast

Summary

```
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyperparameters(ALPHA)", "Test AUC"]

x.add_row(["BOW", "NAIVE BAYES","(0.5)", 0.692])
x.add_row(["TFIDF", "NAIVE BAYES", "(0.05)",  0.690])
print(x)
```

```
+------------+-------------+------------------------+----------+
| Vectorizer |    Model    | Hyperparameters(ALPHA) | Test AUC |
+------------+-------------+------------------------+----------+
|    BOW     | NAIVE BAYES |         (0.5)          |  0.692   |
|   TFIDF    | NAIVE BAYES |         (0.05)         |   0.69   |
+------------+-------------+------------------------+----------+
```

5568	-5.275296	ivan
2421	-5.292911	complex
183	-5.315414	civics_government_visualarts
1857	-5.333552	calendar
1724	-5.341014	brighten
10263	-5.383161	unsafe