DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in nee
volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoos

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three m

- How to scale current manual processes and resources to screen 500,000 projects so that they can be pos
- How to increase the consistency of project vetting across different volunteers to improve the experience f
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a tea
descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then u
to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Descriptio
`project_id`	A unique identifier for the proposed project. **Example:** `p0365`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the p • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **E**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for t • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Exan** • `My students need hands on literacy mate`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*

Feature	Descriptic
project_submitted_datetime	Datetime when project application was submitted. **Example:**
teacher_id	A unique identifier for the teacher of the proposed project. **Ex**
teacher_prefix	Teacher's title. One of the following enumerated values: • nan • Dr. • Mr. • Mrs. • Ms. • Teacher.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the s

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

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Ea a project:

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

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you project:

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

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the projec

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about th school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' lear

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 an

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

```
!ls drive/'My Drive'/data/train_data.csv
```

```
'drive/My Drive/data/train_data.csv'
Mounted at /content/drive
```

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

import pandas as pd
import numpy as np
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc


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


from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os


from collections import Counter
```

1.1 Reading Data

```
project_data = pd.read_csv('drive/My Drive/data/train_data.csv')
resource_data = pd.read_csv('drive/My Drive/data/resources.csv')
```

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

```
print('The Columns with their nan values counts are below ')
for col in project_data.columns:
    print('{col} '.format(col=col),project_data[col].isnull().sum())
```

```
The Columns with their nan values counts are below
Unnamed: 0  0
id  0
teacher_id  0
teacher_prefix  3
school_state  0
project_submitted_datetime  0
project_grade_category  0
project_subject_categories  0
project_subject_subcategories  0
project_title  0
project_essay_1  0
project_essay_2  0
project_essay_3  105490
project_essay_4  105490
project_resource_summary  0
teacher_number_of_previously_posted_projects  0
project_is_approved  0
```

```
# removing 3 nan values from teacher prefix column as they seems to be outliers
# DataFrame.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
project_data.dropna(subset=['teacher_prefix'],inplace=True)
```

```
#how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

```
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

1.2 preprocessing of `project_subject_categories`

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/408

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care &
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scie
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Scie
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True) #0 for index and 1 for co

#counting the occurence of word

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

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

```
# checking the items present category list
cat_dict.items()
```

```
dict_items([('Math_Science', 41419), ('SpecialNeeds', 13642), ('Literacy_Langua
```

1.3 preprocessing of `project_subject_subcategories`

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/408

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care &
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scie
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Scie
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 Text preprocessing

```
# 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)
```

```
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classroom as
==================================================
I teach high school English to students with learning and behavioral disabilit
==================================================
\"Life moves pretty fast. If you don't stop and look around once in awhile, you
==================================================
Some of my students come from difficult family lives, but they don't let that s
==================================================
\"This is how mathematicians do it! Remember we are all mathematicians in this
==================================================
```

```
# 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"\'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)
```

```
Some of my students come from difficult family lives, but they do not let that
==================================================
```

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

```
Some of my students come from difficult family lives, but they do not let that
```

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

```
Some of my students come from difficult family lives but they do not let that s
```

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

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
```

```
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109245/109245 [00:57<00:00, 1883.80it/s]
```

```
# after preprocesing
preprocessed_essays[20000]
```

```
'students come difficult family lives not let stop built community classroom a
```

1.4 Preprocessing of `project_title`

```
# similarly you can preprocess the titles also
preprocessed_title = []

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

```
100%|██████████| 109245/109245 [00:02<00:00, 43334.74it/s]
```

1.5 Preprocessing of `Teacher Prefix`

As the teacher prefix has period assciated with the title like mr.,dr.,etc.

```
# let's check the distribution of this prefix with having period and  special characters
# https://www.geeksforgeeks.org/python-program-check-string-contains-special-character/
import re
regex = re.compile('[@_!#$%^&*()<>?/\|}{~:.]')
project_data.teacher_prefix.map(lambda x: regex.search(x)== None).value_counts()
```

```
False    106885
True       2360
Name: teacher_prefix, dtype: int64
```

There 106885 prefix having period at the end and 2360 titles won't have period. This can be lead to lead to vector differently for features vector.

```
# https://stackoverflow.com/questions/50444346/fast-punctuation-removal-with-pandas
# cleaning the teacher prefix columns as the cells have periods associated with the value like dr
# python's str.translate function is implemented in C, and is therefore very fast.

def clean_col(col):
```

```
    import string
    punct = '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{}~'   # `|` is not present in teacher prefx
    transtab = str.maketrans(dict.fromkeys(punct, ''))

    col = '|'.join(col.tolist()).translate(transtab).split('|')
    return col
```

```
preprocessed_prefix = clean_col(project_data['teacher_prefix'])
```

```
# verifying if any special char are present or not
c = list(map(lambda x : regex.search(x) == None,preprocessed_prefix)).count(True)
print(c)
```

```
109245
```

No special character is present as we have 109245 number of data where all returning true to above regex mean

1.6 Preparing data for models

```
project_data.columns
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', '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',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data (optinal)

- quantity : numerical (optinal)
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

Assignment 4: Apply Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

2. **The hyper paramter tuning(find best Alpha)**
    - Find the best hyper parameter which will give the maximum AUC value
    - Consider a wide range of alpha values for hyperparameter tuning, start as low as 0.00001
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hy

3. **Feature importance**
    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets parameter of MultinomialNB and print their corresponding feature names

4. **Representation of results**
    - You need to plot the performance of model both on train data and cross validation data for each hyp axis you will have alpha values, since they have a wide range, just to represent those alpha values or values.
    - Once after you found the best hyper parameter, you need to train your model with it, and find the AU and test.
    - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original lab confusion matrices using seaborn heatmaps.

5. **Conclusion**
    - You need to summarize the results at the end of the notebook, summarize it in the table format. To library link

2. Naive Bayes

2.1 Splitting data into Train and cross validation(or test): Stratified Sampli

Pandas Dataframe Reordering

Reordering the pandas dataframe with pre processed essays,title and relevant columns for classification

```
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	

```
# checking the aggregate price per resource
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

```
project_data.columns
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', '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',
       'clean_categories', 'clean_subcategories', 'essay', 'price',
       'quantity'],
      dtype='object')
```

```
# https://stackoverflow.com/questions/45747589/copying-a-column-from-one-dataframe-to-another-giv

project_data_ = pd.DataFrame({c: project_data[c].to_numpy() for c in ('school_state','project_gra
                                                                      'teacher_number_of_previousl
project_data_.head(3)
```

	school_state	project_grade_category	clean_categories	clean_subcat
0	CA	Grades PreK-2	Math_Science	AppliedSciences Health_L
1	UT	Grades 3-5	SpecialNeeds	Spe
2	CA	Grades PreK-2	Literacy_Language	

```
# assigning the text data to existing dataframe
project_data_ = project_data_.assign(teacher_prefix = preprocessed_prefix,\
                                     essay = preprocessed_essays,title = preprocessed_title,appro
project_data_.head(3)
```

	school_state	project_grade_category	clean_categories	clean_subcategories
0	CA	Grades PreK-2	Math_Science	AppliedSciences Health_LifeScience
1	UT	Grades 3-5	SpecialNeeds	SpecialNeeds
2	CA	Grades PreK-2	Literacy_Language	Literacy

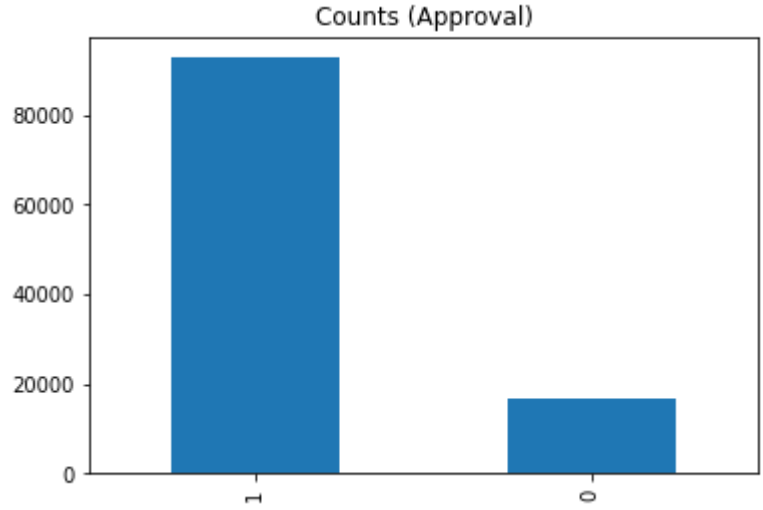
Creating the dataframe for new features first.

```
# Before splitting model let's check if dataset is balanced or not.
print("Negative reviews count = ",np.sum(project_data_.approved==0))
print("positive reviews count = ",np.sum(project_data_.approved==1))
```

```
Negative reviews count =  16542
positive reviews count =  92703
```

```
# vizualing the distribution of class attribute
project_data_.approved.value_counts().plot(kind='bar',title='Counts (Approval)')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f47ad5efe80>
```



As we can clearly see that this dataset higly imbalanced towards positive reviews that means most of the postec

```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = None,None,None,None # clearing the variables
# splitting of train and test data with 80:20 ratio

x_train,x_test,y_train,y_test = train_test_split(project_data_.iloc[:,:project_data_.shape[1]-1],
#X_train,X_cv,y_train,y_cv = train_test_split(X_traincv,y_traincv,test_size=.2,stratify=y_traincv
x_train,x_cv,y_train,y_cv = train_test_split(x_train,y_train,test_size=.2,stratify=y_train)


print("Train Data shape : ",x_train.shape, y_train.shape)
print("Cross Validation Data shape :", x_cv.shape, y_cv.shape)
print("Test Data shape :", x_test.shape, y_test.shape)

print("="*100)
```

```
Train Data shape :  (69916, 9) (69916,)
Cross Validation Data shape : (17480, 9) (17480,)
Test Data shape : (21849, 9) (21849,)
====================================================================================================
```

```
# Count of positive and negative class label in training, cross validation and test data

print("x train distribution = \n", y_train.value_counts())
print("x cv distribution = \n", y_cv.value_counts())
print("x test distribution = \n", y_test.value_counts())
```

```
x train distribution =
 1    59329
0    10587
Name: approved, dtype: int64
x cv distribution =
 1    14833
0     2647
Name: approved, dtype: int64
x test distribution =
 1    18541
0     3308
Name: approved, dtype: int64
```

```
x_train.head(3)
```

	school_state	project_grade_category	clean_categories	clean_subcategor
13801	PA	Grades PreK-2	AppliedLearning	EarlyDevelopn
32710	MA	Grades 9-12	Literacy_Language Math_Science	ESL Mathema
5778	WV	Grades PreK-2	Math_Science	Health_LifeScie

Functions Declaration:

Declaration of functions for which is further used in computational process like

- Vectorization
- Hyperparamater Tuning
- Model Generalisation score on Test Data
- Printing Dimenionality info of input matrix list
- Retrive the vocabulary words for vectorization purposes

```
def retreive_vocab(_data=None):
    ls = []
    for word in _data:
        if len(word) != 1:
            for w in word.split():
                ls.append(w)
        else:
            ls.append(word)
    return list(set(ls))
```

```
# Defining a function to compute BOW, TFIDF

def vectorize_text(encoding_type=None,**kwargs):
    # Validation for proper argument names
    try:
        checklist = ['test_text','cv_text','train_text']
        for k,v in kwargs.items():
            if k in checklist:
                checklist.remove(k)
        if not checklist:
            # do nothing
            pass
        else:
            raise ValueError("You haven't passed the matrices in the described format, please use
```

```
    except ValueError as e:
        print("Error : ", e)

    text_train = kwargs['train_text']
    text_cv = kwargs['cv_text']
    text_test = kwargs['test_text']

    if "BOW" in encoding_type.upper():
        #Compute BOW
        # We are considering only the words which appeared in at least 10 documents(rows or proje
        vectorizer = CountVectorizer(min_df=10,max_features=3000)
        vectorizer.fit(text_train)

        return vectorizer.transform(text_train),vectorizer.transform(text_cv),vectorizer.transfor

    elif "TFIDF" in encoding_type.upper():
        #Compute TFIDF
        from sklearn.feature_extraction.text import TfidfVectorizer
        vectorizer = TfidfVectorizer(min_df=10,max_features=3000)
        vectorizer.fit(text_train)
        return vectorizer.transform(text_train),vectorizer.transform(text_cv),vectorizer.transfor



    else:
        raise ValueError('Please give the encoding type from the following: BOW, TFIDF, AVGW2V,TF
```

```
def _hypertuning(x_train,y_train,x_cv,y_cv,tune_type):
    _alpha = [0.00001,0.00005,0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1,1.5,2,2.5,3,3.5,4,4.5
    from sklearn.naive_bayes import MultinomialNB

    if tune_type.lower() == 'custom':

        from sklearn.metrics import roc_auc_score

        #y_train_pred = []
        train_auc_score = []
        cv_auc_score = []
        e
        for i in _alpha:
            y_train_pred = []
            y_cv_pred = []
            clf = MultinomialNB(alpha = i)
            clf.fit(x_train,y_train)

        # return value of predict_proba : array of shape = [n_samples, n_classes], or a list of n
        # since the value of probabilities obtained will be much less, let's consider the log pro
            y_train_pred.extend(clf.predict_log_proba(x_train)[:,1])
            y_cv_pred.extend(clf.predict_log_proba(x_cv)[:,1])
            #import pdb
            #pdb.set_trace()
            train_auc_score.append(roc_auc_score(y_train,y_train_pred))
            cv_auc_score.append(roc_auc_score(y_cv,y_cv_pred))



        plt.plot(np.log(_alpha), train_auc_score, label='Train AUC')
        plt.scatter(np.log(_alpha),train_auc_score)
```

```
        plt.scatter(np.log(_alpha),train_auc_score)
        plt.plot(np.log(_alpha), cv_auc_score, label='CV AUC')
        plt.scatter(np.log(_alpha),cv_auc_score)
        plt.legend()
        plt.xlabel("log alpha: hyperparameter")
        plt.ylabel("AUC")
        plt.title("ERROR PLOTS")
        plt.show()



    elif tune_type.lower() == 'gridsearch':

        # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.
        from sklearn.model_selection import GridSearchCV

        nb_clf = MultinomialNB()
        parameters = {'alpha' :[0.00001,0.00005,0.0001,0.0005,0.001,0.005,0.01,0.05,0.1,0.5,1,1.5
        clf = GridSearchCV(nb_clf, parameters, cv=10, scoring='roc_auc',n_jobs=4,return_train_sco
        clf.fit(x_train, 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']

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

        plt.plot(np.log(parameters['alpha']), cv_auc, label='CV AUC')
        # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
        plt.gca().fill_between(np.log(parameters['alpha']),cv_auc - cv_auc_std,cv_auc + cv_auc_st
        plt.legend()
        plt.xlabel("log alpha: hyperparameter")
        plt.ylabel("AUC")
        plt.title("ERROR PLOTS")
        plt.show()
```

```
#https://www.ritchieng.com/machine-learning-evaluate-classification-model/

def evaluate_threshold(_alpha,x_train,y_train,x_cv,y_cv):
    from sklearn.naive_bayes import MultinomialNB
    y_cv_pred = []
    nb_clf = MultinomialNB(alpha = _alpha)
    nb_clf.fit(x_train,y_train)
    y_cv_pred.extend(nb_clf.predict_log_proba(x_cv)[:,1])
    from sklearn.metrics import roc_curve
    #import pdb
    #pdb.set_trace()
    fpr,tpr,thresholds = roc_curve(y_true = y_cv,y_score = y_cv_pred)
    #t_val = [0.1,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65,0.7,0.8,0.9] # different probabili
    t_val = np.arange(0,1,0.05)
    sn,sp =0,0
```

```
    ss_score = list()
    for i in np.log(t_val):
        sn = tpr[thresholds > i][-1]
        sp = 1 - fpr[thresholds > i][-1]
        ss_score.append((sn,sp,i))
    from prettytable import PrettyTable
    x = PrettyTable()
    x.field_names = ['Sensitivity/Recall (TPR)','Specificity (1-FPR)','Threshold Value','Log Prob
    for val in ss_score:
        sn,sp,th = val
        x.add_row([sn,sp,np.exp(th),th]) # taking anti log here
    print(x)
```

```
# https://stackoverflow.com/questions/19984957/scikit-predict-default-threshold

def model_gen_score(x_train,y_train,x_test,y_test,best_alpha,cutoff_val,features_names):
    from sklearn.metrics import roc_curve, auc
    from sklearn.metrics import confusion_matrix
    from sklearn.naive_bayes import MultinomialNB

    nb_clf = MultinomialNB(alpha=best_alpha)
    nb_clf.fit(x_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positiv
# not the predicted outputs
    y_train_pred_prob = []
    #y_train_pred = []
    y_test_pred_prob = []
    #y_test_pred = []

    y_train_pred_prob.extend(nb_clf.predict_log_proba(x_train)[:,1])
    y_test_pred_prob.extend(nb_clf.predict_log_proba(x_test)[:,1])

    train_fpr, train_tpr, train_thresholds = roc_curve(y_true=y_train,y_score=y_train_pred_prob)
    test_fpr, test_tpr, test_thresholds = roc_curve(y_true=y_test,y_score=y_test_pred_prob)

    plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
    #plt.scatter(train_fpr, np.exp(train_thresholds))
    plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
    #plt.scatter(test_fpr, np.exp(test_thresholds))
    plt.legend()
    plt.xlabel("FPR (1 - Specificity)")
    plt.ylabel("TPR (Sensitivity)")
    plt.title("ROC Curve")
    plt.show()


    # predicting the number of important features
    # https://stackoverflow.com/questions/50526898/how-to-get-feature-importance-in-naive-bayes

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

    print("Negative Class Important Features \n",np.take(features_names, neg_class_prob_sorted[:1
    print("**********************************************************************")
    print("Positive Class Important Features \n",np.take(features_names, pos_class_prob_sorted[:1
```

```
    # Confusion matrix evaluations

    print("="*100)
    y_train_pred = (np.array(y_train_pred_prob) >= cutoff_val).astype(int)
    y_test_pred = (np.array(y_test_pred_prob) >= cutoff_val).astype(int)
    f, (ax1, ax2) = plt.subplots(2, 1,figsize=[8,8])
    print("Confusion matrix \n")
    sns.heatmap(data=confusion_matrix(y_train, y_train_pred),annot=True,fmt="",ax=ax1)
    ax1.set_title('Train confusion matrix')
    sns.heatmap(data=confusion_matrix(y_test, y_test_pred),annot=True,fmt="",ax=ax2)
    ax2.set_title('Test confusion matrix')
    plt.show()
    return (auc(train_fpr, train_tpr),auc(test_fpr, test_tpr))
```

```
def print_dimension_info(_obj,_name):
    data_list= ['Training count : ','Cross Validation count : ','Test count : '] * len(_obj)
    col_num = []
    row_num = list()
    for i in _obj:
        row_num.append(i.shape[0])
        col_num.append(i.shape[1])
    print("The Values for : ", _name)
    print("\nRow Values are : ",list(zip(data_list,row_num)))
    print("\nColumn Values are : ",list(zip(data_list,col_num)))
    print("\nType of matrices: ",[type(x) for x in _obj])
    print("*"*100)
```

```
def one_hot_encoder(df_col_train,df_col_cv,df_col_test,vocab=None,case=False,_bin=True):
    encoder_obj = CountVectorizer(vocabulary = vocab,lowercase=case,binary=_bin)
    encoder_obj.fit(df_col_train)
    print("features are : \n",encoder_obj.get_feature_names())

    return encoder_obj.transform(df_col_train.values),encoder_obj.transform(df_col_cv.values),enc
```

2.2 Make Data Model Ready: encoding numerical, categorical features

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

```
# Vectorizing the teacher prefix input
x_train_tp,x_cv_tp,x_test_tp,tp_feat_names = one_hot_encoder(df_col_train=x_train['teacher_prefix
```

```
                                                df_col_cv=x_cv['teacher_prefix'],\
                                                df_col_test=x_test['teacher_prefix'])
```

```
features are :
 ['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
```

```
print(x_train_tp[1:10])
```

```
  (0, 3)        1
  (1, 3)        1
  (2, 2)        1
  (3, 3)        1
  (4, 2)        1
  (5, 2)        1
  (6, 3)        1
  (7, 3)        1
  (8, 1)        1
```

```
# vectorizing the school state column
x_train_ss,x_cv_ss,x_test_ss,ss_feat_names = one_hot_encoder(df_col_train=x_train['school_state']
                                                df_col_test=x_test['school_state'])
```

```
features are :
 ['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA'
```

```
# Vectorizing the project_grade_category
x_train_pgc,x_cv_pgc,x_test_pgc,pgc_feat_names = one_hot_encoder(df_col_train=x_train['project_gr
                                                   df_col_test=x_test['project_grade_category'],vo
```

```
features are :
 ['Grades PreK-2', 'Grades 9-12', 'Grades 6-8', 'Grades 3-5']
```

```
# Vectorizing the project subject category
x_train_cat,x_cv_cat,x_test_cat,cat_feat_names = one_hot_encoder(df_col_train=x_train['clean_cate
                                                   df_col_test=x_test['clean_categories'],vocab=re
```

```
features are :
 ['AppliedLearning', 'Health_Sports', 'Care_Hunger', 'SpecialNeeds', 'Literacy_
```

```
# Vectorizing the project subject sub category
x_train_sub,x_cv_sub,x_test_sub,sub_feat_names = one_hot_encoder(df_col_train=x_train['clean_subc
                                                   df_col_test=x_test['clean_subcategories'],vocab
```

```
features are :
 ['ForeignLanguages', 'NutritionEducation', 'SpecialNeeds', 'Other', 'History_
```

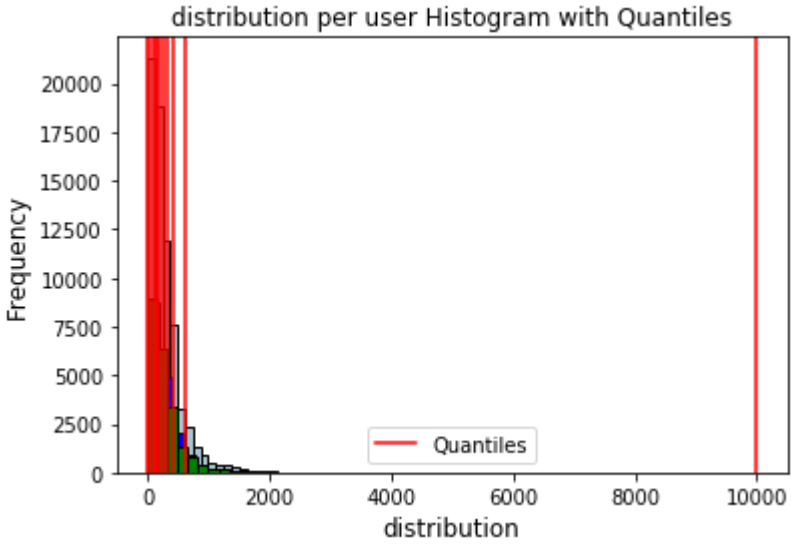
Let's check if the data distribution is close to normal distribution. So we gonna plot the price random variable and will checking if they are close to normal distribution or not. If they are close to normal distribution then we gonna

accurate probablities values. else we might have to bin them and then proceed further with multinomial naive ba

```
# cutting the real valued functions on the basis every tenth percentiles
# Adaptive Binning

def bin_data(s_train,s_cv,s_test):
    # https://towardsdatascience.com/understanding-feature-engineering-part-1-continuous-numeric-

    quantile_list = [0.,.1, .2,.3,.4, .5,.6 ,.7,.8,.9, 1.]
    quantiles = s_train.quantile(quantile_list)
    print(quantiles)
    fig, ax = plt.subplots()
    s_train.hist(bins=80, color='#A9C5D3', edgecolor='black', grid=False)
    s_cv.hist(bins=50,color='blue',edgecolor='black',grid=False)
    s_test.hist(bins=60,color='green',edgecolor='black',grid=False)
    for quantile in quantiles:
        qvl = plt.axvline(quantile, color='r')
    ax.legend([qvl], ['Quantiles'], fontsize=10)
    ax.set_title('distribution per user Histogram with Quantiles',
                 fontsize=12)
    ax.set_xlabel('distribution', fontsize=12)
    ax.set_ylabel('Frequency', fontsize=12)


    s_train = pd.qcut(s_train,q=quantile_list,duplicates='drop')
    s_cv = pd.qcut(s_cv,q=quantile_list,duplicates = 'drop')
    s_test = pd.qcut(s_test,q=quantile_list,duplicates='drop')

    return s_train,s_cv,s_test
```

```
x_train_price,x_cv_price,x_test_price = bin_data(s_train = x_train['price'],\
                                                 s_cv = x_cv['price'],s_test=x_test['price'])
```

```
0.0       0.660
0.1      38.515
0.2      83.980
0.3     123.290
0.4     164.020
0.5     206.910
0.6     266.930
0.7     334.975
0.8     428.680
0.9     619.300
1.0    9999.000
Name: price, dtype: float64
```




```
x_train_price.astype(str).unique()
```

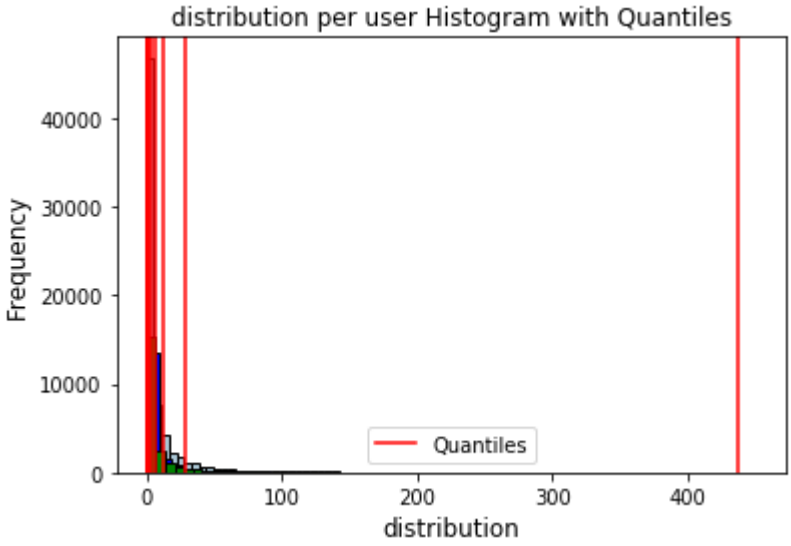
```
array(['(428.68, 619.3]', '(38.515, 83.98]', '(83.98, 123.29]',
       '(619.3, 9999.0]', '(123.29, 164.02]', '(266.93, 334.975]',
       '(0.659, 38.515]', '(164.02, 206.91]', '(206.91, 266.93]',
       '(334.975, 428.68]'], dtype=object)
```

```
x_train_price,x_cv_price,x_test_price,price_feat_names = one_hot_encoder(df_col_train=x_train_pri
                                                         df_col_cv=x_cv_price.astype(str),\
                                                         df_col_test=x_test_price.astype(str),voca
```

```
features are :
 ['(428.68, 619.3]', '(38.515, 83.98]', '(83.98, 123.29]', '(619.3, 9999.0]',
```

```
x_train_pp_count,x_cv_pp_count,x_test_pp_count = bin_data(s_train = x_train['teacher_number_of_pr
                                                          s_cv = x_cv['teacher_number_of_previousl
                                                          s_test = x_test['teacher_number_of_previ
                                                          )
```

```
0.0      0.0
0.1      0.0
0.2      0.0
0.3      1.0
0.4      1.0
0.5      2.0
0.6      4.0
0.7      6.5
0.8     12.0
0.9     28.0
1.0    437.0
Name: teacher_number_of_previously_posted_projects, dtype: float64
```




```
x_train_pp_count.astype(str).unique()
```

```
array(['(-0.001, 1.0]', '(1.0, 2.0]', '(28.0, 437.0]', '(2.0, 4.0]',
       '(4.0, 6.5]', '(12.0, 28.0]', '(6.5, 12.0]'], dtype=object)
```

```
x_train_pp_count,x_cv_pp_count,x_test_pp_count,pp_feat_names = one_hot_encoder(df_col_train=x_tra
                                                               df_col_cv=x_cv_pp_count.astype(st
                                                               df_col_test=x_test_pp_count.astyp
                                                               vocab=x_train_pp_count.astype(str
```

```
features are :
 ['(-0.001, 1.0]', '(1.0, 2.0]', '(28.0, 437.0]', '(2.0, 4.0]', '(4.0, 6.5]',
```

```
# printing the dimensions of vectorized of dataset
print_dimension_info(_obj=[x_train_tp,x_cv_tp,x_test_tp],_name='Teacher Prefix')
print_dimension_info(_obj=[x_train_ss,x_cv_ss,x_test_ss],_name='Schol State Column')
print_dimension_info(_obj=[x_train_pgc,x_cv_pgc,x_test_pgc],_name = 'Project Grade Category')
print_dimension_info(_obj=[x_train_cat,x_cv_cat,x_test_cat],_name= 'Project subject category')
print_dimension_info(_obj=[x_train_sub,x_cv_sub,x_test_sub],_name= 'Project subject sub category'
print_dimension_info(_obj=[x_train_price,x_cv_price,x_test_price],_name = 'Price')
print_dimension_info(_obj=[x_train_pp_count,x_cv_pp_count,x_test_pp_count],_name='Count of previc
```

```
The Values for :  Teacher Prefix

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 5), ('Cross Validation count : ', 5

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**************************************************************************
The Values for :  Schol State Column

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 51), ('Cross Validation count : ',

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**************************************************************************
The Values for :  Project Grade Category

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 4), ('Cross Validation count : ', 4

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**************************************************************************
The Values for :  Project subject category

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 9), ('Cross Validation count : ', 9

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**************************************************************************
The Values for :  Project subject sub category

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 30), ('Cross Validation count : ',

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**************************************************************************
The Values for :  Price

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 10), ('Cross Validation count : ',

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**************************************************************************
The Values for :  Count of previous project submitted by teacher

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 7), ('Cross Validation count : ', 7

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**************************************************************************
```

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we can concatenate a sparse matrix and a dense matrix :)
```

```
# with the same hstack function we can concatenate a sparse matrix and a dense matirx :)
x_train_vec = hstack((x_train_tp, x_train_ss, x_train_pgc, x_train_cat, x_train_sub, x_train_pric
x_cv_vec = hstack((x_cv_tp, x_cv_ss, x_cv_pgc, x_cv_cat, x_cv_sub, x_cv_price, x_cv_pp_count),for
x_test_vec = hstack((x_test_tp, x_test_ss, x_test_pgc, x_test_cat, x_test_sub, x_test_price, x_te

print_dimension_info(_obj=[x_train_vec,x_cv_vec,x_test_vec],_name='Stacked sparse matrices dimens
```

```
The Values for :  Stacked sparse matrices dimensions

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 116), ('Cross Validation count : '

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**********************************************************************
```

2.3 Make Data Model Ready: encoding essay, and project_title

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

```
# Bag of words vectorization for train,cv and test data
bow_title_train,bow_title_cv,bow_title_test,bow_title_features = vectorize_text(encoding_type='BO
                                                                cv_text =x_cv['title'] , test_text =

print_dimension_info(_obj=[bow_title_train,bow_title_cv,bow_title_test],_name='Dimensions after B

bow_essay_train,bow_essay_cv,bow_essay_test,bow_essay_features = vectorize_text(encoding_type='BO
                                                                cv_text =x_cv['essay'] , test_text =
print_dimension_info(_obj=[bow_essay_train,bow_essay_cv,bow_essay_test],_name='Dimensions after B
```

```
The Values for :  Dimensions after BOW on Title :

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 2469), ('Cross Validation count :

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**********************************************************************
The Values for :  Dimensions after BOW on essay

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 3000), ('Cross Validation count :

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**********************************************************************
```

```
# Stacking all the BOW models with existing data frame -

final_x_train = hstack((x_train_vec,bow_title_train,bow_essay_train),format='csr')
final_x_cv = hstack((x_cv_vec,bow_title_cv,bow_essay_cv),format='csr')
final_x_test = hstack((x_test_vec,bow_title_test,bow_essay_test),format='csr')

# stacking all their features also so that we can interpret the features importance here.
stacked_feature_list = [tp_feat_names,ss_feat_names,pgc_feat_names,cat_feat_names,sub_feat_names,
                ['price range '+x for x in price_feat_names],['pp count '+x for x in pp_feat_name
feature_list = list()
for features in stacked_feature_list:
    feature_list.extend(features)
#print("All general features are: \n",feature_list)
# stacking for features for bow model
bow_feature_list = list()
bow_feature_list.extend(feature_list)
for features in [bow_title_features,bow_essay_features]:
    bow_feature_list.extend(features)

print_dimension_info(_obj=[final_x_train,final_x_cv,final_x_test],_name='The Final Matrix dimensi
```

```
The Values for :  The Final Matrix dimension info after BOW

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 5585), ('Cross Validation count :

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**********************************************************************
```

2.4 Appling NB() on different kind of featurization as mentioned in the inst

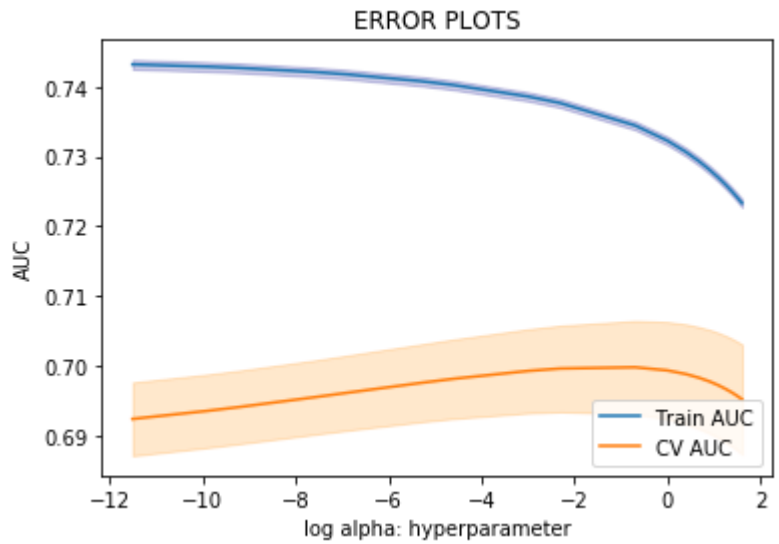
Apply Naive Bayes on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do
```

```
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code.

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

2.4.1 Applying NB brute force on BOW, SET 1

```
# cross validation with BOW model

_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv=y_cv.values,tune
```

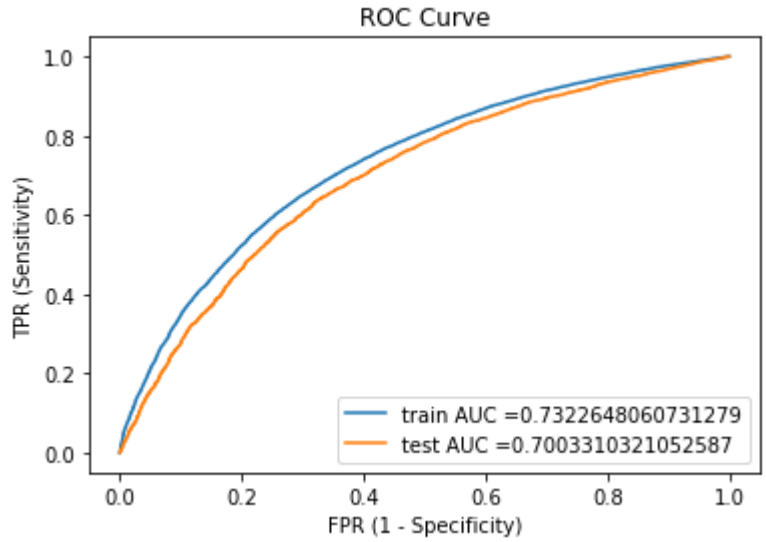



```
# Deciding appropriate threshold with cross validation dataset
evaluate_threshold(_alpha=np.exp(-1),x_train = final_x_train,y_train=y_train.values,x_cv=final_x_
```

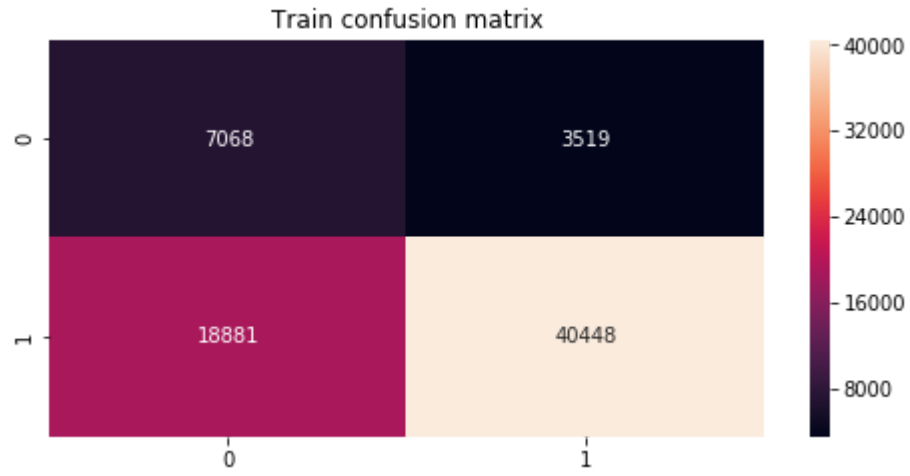
Sensitivity/Recall (TPR)	Specificity (1-FPR)	Threshold Value	Log P
1.0	0.0	0.0	
0.890649228072541	0.29580657347941064	0.05000000000000001	-2
0.8623339850333716	0.3494522100491122	0.10000000000000002	-2
0.8430526528686038	0.3849641103135625	0.15000000000000002	-1
0.8291647003303445	0.4064979221760484	0.2	-1
0.8152767477920853	0.4238760861352474	0.25	-1
0.8035461471044293	0.44276539478655086	0.30000000000000004	-1
0.7920852153980988	0.45787684170759346	0.35000000000000003	-1
0.7808265354277624	0.4726105024556101	0.4	-0
0.768624014022787	0.48885530789573106	0.45	-0
0.7584440099777523	0.5020778239516434	0.5	-0
0.7449605609114811	0.5187004155647903	0.55	-0
0.7320164498078608	0.5292784284095202	0.6000000000000001	-0
0.7203532663655363	0.5443898753305629	0.65	-0
0.7086226656778803	0.5644125425009445	0.7000000000000001	-0
0.6927121957796805	0.5821684926331696	0.75	-0
0.6759253016921729	0.5972799395542123	0.8	-0
0.6534079417515001	0.6233471854930109	0.8500000000000001	-0
0.6207779950111239	0.6535700793350963	0.9	-0
0.5701476437672757	0.7057045712126936	0.9500000000000001	-0

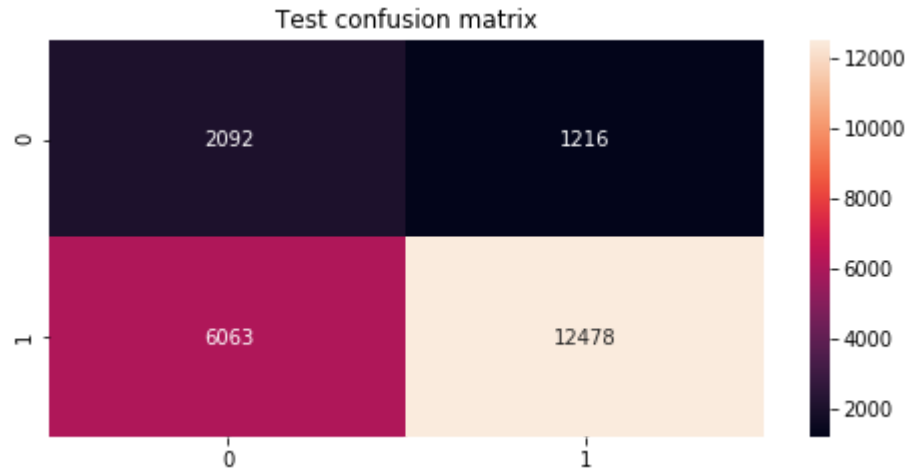
In general, We will be choosing Threshold value as 0.8 (log threshold = -0.22) as it maximizes Sensitivity as well a
values means relative less false negatives
Also it depends on the case where what we want to maximize either TPR (or minimise FN) or FPR (or maximize F
For Alpha we will be taking values around log threshold -1. As post this value there is not much growth in cross v
decrease drastically.

```
# Checking the auc score with proper K value
bow_aplha = np.exp(-1)
bow_train_auc,bow_test_auc = model_gen_score(x_train = final_x_train,y_train=y_train.values,\
                                             x_test=final_x_test,y_test=y_test.values,best_alpha=
                                            cutoff_val= -0.22,features_names=bow_feature_list)
```




```
Negative Class Important Features
 ['magna' 'choir' 'textbooks' '60' 'bikes' 'ten' 'pokey' 'teenagers'
 'search' 'easels']
**************************************************************************
Positive Class Important Features
 ['Grades 3-5' 'pp count (28.0, 437.0]' 'pp count (1.0, 2.0]'
 'pp count (-0.001, 1.0]' 'price range (334.975, 428.68]'
 'price range (206.91, 266.93]' 'price range (164.02, 206.91]'
 'Grades 6-8' 'price range (0.659, 38.515]'
 'price range (266.93, 334.975]']
==========================================================================
Confusion matrix
```





The Test Accuracy (70%) is almost similar to train Accuracy (73%) which is quite good as this model is generalizi

2.4.2 Applying NB brute force on TFIDF, SET 2

```
# Please write all the code with proper documentation

tfidf_essay_train,tfidf_essay_cv,tfidf_essay_test,tfidf_essay_features = vectorize_text(encoding_
                                                                        cv_text =x_cv['essay'] , test_text =
```

```
# Please write all the code with proper documentation

tfidf_title_train,tfidf_title_cv,tfidf_title_test,tfidf_title_features = vectorize_text(encoding_
                                                                        cv_text =x_cv['title'] , test_text =
```

```
final_x_train = hstack((x_train_vec,tfidf_title_train,tfidf_essay_train),format='csr')
final_x_cv = hstack((x_cv_vec,tfidf_title_cv,tfidf_essay_cv),format='csr')
final_x_test = hstack((x_test_vec,tfidf_title_test,tfidf_essay_test),format='csr')

tfidf_feature_list = list()
tfidf_feature_list.extend(feature_list)
#print('tfidf features ', tfidf_feature_list)
for features in [tfidf_title_features,tfidf_essay_features]:
    tfidf_feature_list.extend(features)

print_dimension_info(_obj=[final_x_train,final_x_cv,final_x_test],_name='The Final Matrix dimensi
```

```
The Values for :  The Final Matrix dimension info after TFIDF

Row Values are :  [('Training count : ', 69916), ('Cross Validation count : ',

Column Values are :  [('Training count : ', 5585), ('Cross Validation count :

Type of matrices:  [<class 'scipy.sparse.csr.csr_matrix'>, <class 'scipy.sparse
**********************************************************************************
```

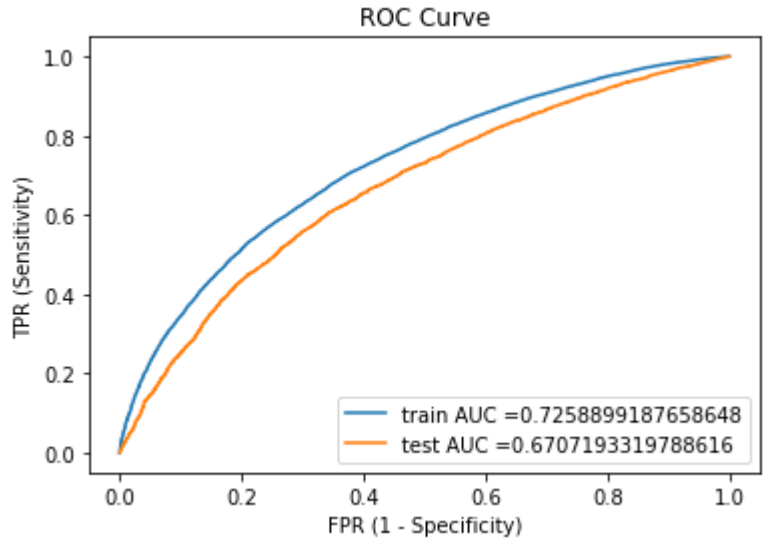
```
# Hypertuning the parameters for cross validation

_hypertuning(x_train = final_x_train,y_train=y_train.values,x_cv=final_x_cv,y_cv=y_cv.values,tune
```

```
# Deciding appropriate threshold with cross validation dataset
evaluate_threshold(_alpha=np.exp(-0.8),x_train = final_x_train,y_train=y_train.values,x_cv=final_
```

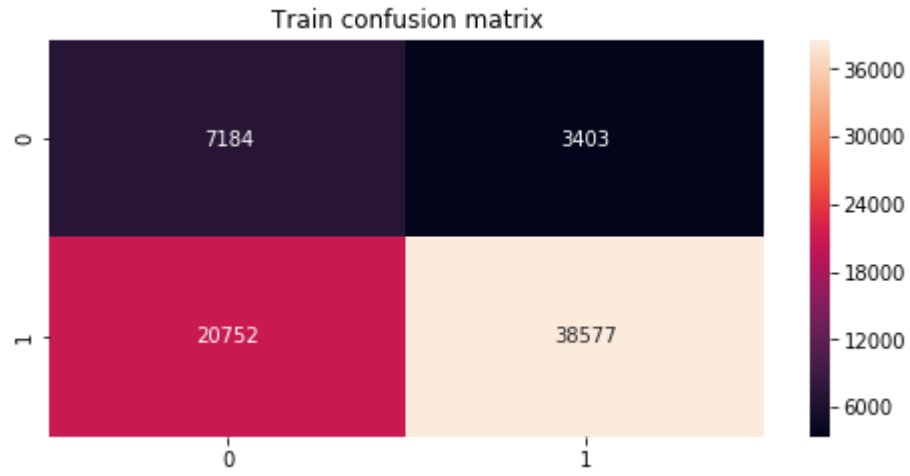
```
+---------------------------+------------------------+----------------------+-----
| Sensitivity/Recall (TPR) |  Specificity (1-FPR)  |   Threshold Value    | Log
+---------------------------+------------------------+----------------------+-----
|           1.0             |          0.0           |         0.0          |
|           1.0             |          0.0           | 0.05000000000000001  |
|           1.0             |          0.0           | 0.10000000000000002  |
|           1.0             |          0.0           | 0.15000000000000002  |
|           1.0             |          0.0           |         0.2          |
|    0.9999325827546687     |          0.0           |         0.25         |
|    0.9993932447920177     |  0.000755572346052169  | 0.30000000000000004  |
|    0.9987864895840356     | 0.0037778617302606232  | 0.35000000000000003  |
|    0.9971010584507517     |  0.008311295806573527  |         0.4          |
|    0.9942021169015034     |  0.018889308651303338  |         0.45         |
|    0.9877300613496932     |  0.03362296939931997   |         0.5          |
|    0.9786287332299602     |  0.06233471854930106   |         0.55         |
|    0.9633250185397425     |  0.11106913486966374   |  0.6000000000000001  |
|    0.9375716308231645     |   0.1711371363808084   |         0.65         |
|    0.8987392975123036     |  0.24367208160181342   |  0.7000000000000001  |
|    0.8402885458100182     |   0.3313184737438609   |         0.75         |
|    0.758376592732421      |   0.455232338496411    |         0.8          |
|    0.6408009168745366     |   0.5931242916509256   |  0.8500000000000001  |
|    0.4601901166318344     |   0.7669059312429165   |         0.9          |
|   0.18324007281062496     |   0.9308651303362296   |  0.9500000000000001  |
+---------------------------+------------------------+----------------------+-----
```

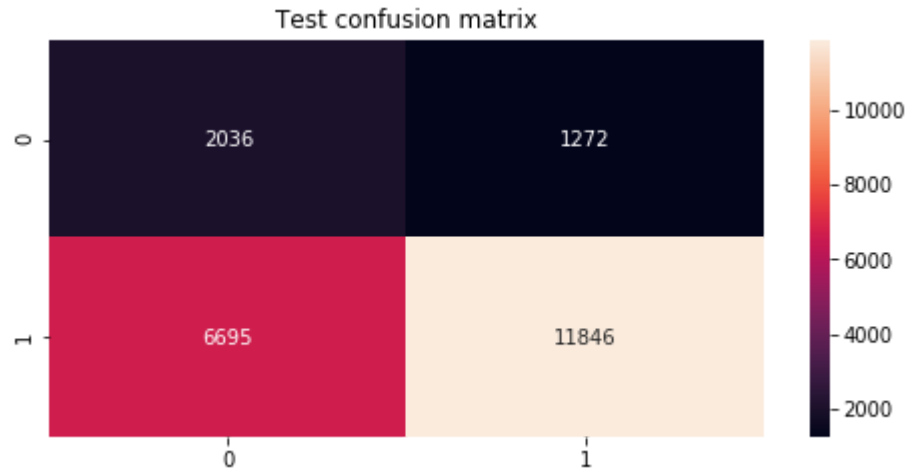
There is certain spikes between -1 and 0 values. In general, We will be choosing Threshold value as 0.85 (log thre
Specificity. Also it's more inclined towards TPR values means relative less false negatives
Also it depends on the case where what we want to maximize either TPR (or minimise FN) or FPR (or maximize F
For Alpha we will be taking values around log threshold -0.8. As post this value there is not much growth in cross
decrease drastically.

```
# Testing the data on Test data to check generalization of this Model
tfidf_alpha = np.exp(-0.8)
tfidf_train_auc,tfidf_test_auc = model_gen_score(x_train = final_x_train,y_train = y_train.values
```




```
Negative Class Important Features
 ['pp count (2.0, 4.0]' 'pp count (6.5, 12.0]' '000' 'donate' 'meaning'
 'maximizing' 'zenergy' 'thirst' 'mat' 'pp count (12.0, 28.0]']
**************************************************************************
Positive Class Important Features
 ['pp count (4.0, 6.5]' 'pp count (2.0, 4.0]' 'pp count (28.0, 437.0]'
 'pp count (1.0, 2.0]' 'pp count (-0.001, 1.0]'
 'price range (334.975, 428.68]' 'price range (206.91, 266.93]'
 'price range (164.02, 206.91]' 'pp count (12.0, 28.0]'
 'price range (266.93, 334.975]']
==========================================================================
Confusion matrix
```





The train AUC i.e. 72% and Test AUC i.e. 67% is almost closer but BOW model was generalizing well as compared
Also The false negatives are relatively lesser as compared to other values(FP,TN,TP)

3. Conclusions

```
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ['Vectorizer','Model','Hyperparameters','TEST AUC']

x.add_row(['BOW','BRUTE',bow_aplha,bow_test_auc])
x.add_row(['TFIDF','BRUTE',tfidf_alpha,tfidf_test_auc])

print(x)
```

```
+------------+-------+----------------------+--------------------+
| Vectorizer | Model |   Hyperparameters    |      TEST AUC      |
+------------+-------+----------------------+--------------------+
|    BOW     | BRUTE | 0.36787944117144233 | 0.7003310321052587 |
|   TFIDF    | BRUTE | 0.44932896411722156 | 0.6707193319788616 |
+------------+-------+----------------------+--------------------+
```

- The BOW model has better generalized with the AUC of 70% while TFIDF model (AUC 67%).
- Since, Real valued random variables in this dataset were following pareto Distribution, instead of Gaussian variable data to use multinomial NB directly on the whole dataset so that we do hyper parameter tuning or
- Also binning can remove the noise from data.