# Importing Libraries

In [1]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
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
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

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

from tqdm import tqdm
import os

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

from collections import Counter
from prettytable import PrettyTable

from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math
```

# Loading data

In [2]:

```
project_data = pd.read_csv('train_data.csv')
resource_data =pd.read_csv('resources.csv')
```

In [3]:

```
project_data.head()
```

Out[3]:

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

In [4]:

```
resource_data.head()
```

Out[4]:

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

In [5]:

```
project_data.isnull().any()
```

Out[5]:

```
Unnamed: 0                                      False
id                                              False
teacher_id                                      False
teacher_prefix                                   True
school_state                                    False
project_submitted_datetime                      False
project_grade_category                          False
project_subject_categories                      False
project_subject_subcategories                   False
project_title                                   False
project_essay_1                                 False
project_essay_2                                 False
project_essay_3                                  True
project_essay_4                                  True
project_resource_summary                        False
teacher_number_of_previously_posted_projects    False
project_is_approved                             False
dtype: bool
```

In [6]:

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

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

In [7]:

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

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
        id                                        description  quantity  \
0  p233245  LC652 - Lakeshore Double-Space Mobile Drying Rack         1
1  p069063        Bouncy Bands for Desks (Blue support pipes)         3

    price
0  149.00
1   14.95
```

In [8]:

```
project_data['teacher_prefix']= project_data['teacher_prefix'].fillna(project_data['tea
cher_prefix'].mode().iloc[0])
```

In [9]:

```
# 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)
```

## preprocessing subject category

In [10]:

```
categories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
ng
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
on
cat_list = []
for i in categories:
    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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
ces
        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)

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]))
```

## preprocessing subject subcategory

In [11]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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", "Warmt
h", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "M
ath & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace
 it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
ath & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spa
ces
        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]))
```

In [12]:

```
y = project_data['project_is_approved'].values
X = project_data.drop(['project_is_approved','Unnamed: 0','teacher_id','project_submitt
ed_datetime'], axis=1)
X.head(1)
```

Out[12]:

| | id | teacher_prefix | school_state | project_grade_category | project_title | projec |
|---|---|---|---|---|---|---|
| **0** | p253737 | Mrs. | IN | Grades PreK-2 | Educational Support for English Learners at Home | My stu Englis that ar |

## Spliting the data into train and test

In [13]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, strat
ify=y_train)
```

## Text preprocessing

### For Essay

In [14]:

```
# 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
```

In [15]:

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

```
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. The
y are eager beavers and always strive to work their hardest working past t
heir limitations. \r\n\r\nThe materials we have are the ones I seek out fo
r my students. I teach in a Title I school where most of the students rece
ive free or reduced price lunch.  Despite their disabilities and limitatio
ns, my students love coming to school and come eager to learn and explore.
Have you ever felt like you had ants in your pants and you needed to groov
e and move as you were in a meeting? This is how my kids feel all the tim
e. The want to be able to move as they learn or so they say.Wobble chairs
are the answer and I love then because they develop their core, which enha
nces gross motor and in Turn fine motor skills. \r\nThey also want to lear
n through games, my kids do not want to sit and do worksheets. They want t
o learn to count by jumping and playing. Physical engagement is the key to
our success. The number toss and color and shape mats can make that happe
n. My students will forget they are doing work and just have the fun a 6 y
ear old deserves.nannan
==================================================
```

In [16]:

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

```
My kindergarten students have varied disabilities ranging from speech and
language delays, cognitive delays, gross/fine motor delays, to autism. The
y are eager beavers and always strive to work their hardest working past t
heir limitations.     The materials we have are the ones I seek out for my
students. I teach in a Title I school where most of the students receive f
ree or reduced price lunch.  Despite their disabilities and limitations, m
y students love coming to school and come eager to learn and explore.Have
you ever felt like you had ants in your pants and you needed to groove and
move as you were in a meeting? This is how my kids feel all the time. The
want to be able to move as they learn or so they say.Wobble chairs are the
answer and I love then because they develop their core, which enhances gro
ss motor and in Turn fine motor skills.   They also want to learn through
games, my kids do not want to sit and do worksheets. They want to learn to
count by jumping and playing. Physical engagement is the key to our succes
s. The number toss and color and shape mats can make that happen. My stude
nts will forget they are doing work and just have the fun a 6 year old des
erves.nannan
```

In [17]:

```
#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 delays cognitive delays gross fine motor delays to autism They ar
e eager beavers and always strive to work their hardest working past their
limitations The materials we have are the ones I seek out for my students
I teach in a Title I school where most of the students receive free or red
uced price lunch Despite their disabilities and limitations my students lo
ve coming to school and come eager to learn and explore Have you ever felt
like you had ants in your pants and you needed to groove and move as you w
ere in a meeting This is how my kids feel all the time The want to be able
to move as they learn or so they say Wobble chairs are the answer and I lo
ve then because they develop their core which enhances gross motor and in
Turn fine motor skills They also want to learn through games my kids do no
t want to sit and do worksheets They want to learn to count by jumping and
playing Physical engagement is the key to our success The number toss and
color and shape mats can make that happen My students will forget they are
doing work and just have the fun a 6 year old deserves nannan
```

In [18]:

```
# 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'r
e", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 't
hey', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "th
at'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
d', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as'
, 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through'
, 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'ov
er', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
y', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too'
, 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'no
w', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'migh
tn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'w
asn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [19]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_train_essay = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['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_train_essay.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████
████| 49041/49041 [01:00<00:00, 806.06it/s]
```

In [20]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_cv_essay = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['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_cv_essay.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████
████| 24155/24155 [00:29<00:00, 812.09it/s]
```

In [21]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_test_essay = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['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_test_essay.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████
████| 36052/36052 [00:45<00:00, 790.34it/s]
```

## For Title

In [22]:

```
# 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
```

In [23]:

```
sent = decontracted(project_data['project_title'].values[37000])
print(sent)
print('='*50)
```

```
Focus our CORE!
==================================================
```

In [24]:

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

```
Focus our CORE!
```

In [25]:

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

```
Focus our CORE
```

In [26]:

```
# 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'r
e", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 't
hey', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "th
at'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
d', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as'
, 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through'
, 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'ov
er', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
y', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too'
, 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'no
w', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'migh
tn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'w
asn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [27]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_train_title = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['project_title'].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_train_title.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████
████| 49041/49041 [00:02<00:00, 16959.77it/s]
```

In [28]:

```
preprocessed_cv_title = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['project_title'].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_cv_title.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████
████| 24155/24155 [00:01<00:00, 17014.81it/s]
```

In [29]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_test_title = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['project_title'].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_test_title.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████
████| 36052/36052 [00:02<00:00, 17134.24it/s]
```

## One hot encoding

### Categories

In [30]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vec1 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), binary=True)
X_train_cat_ohe = vec1.fit_transform(X_train['clean_categories'].values)
X_cv_cat_ohe = vec1.transform(X_cv['clean_categories'].values)
X_test_cat_ohe = vec1.transform(X_test['clean_categories'].values)

print("After Vectorizations")
print("Shape of X_train after one hot encodig ",X_train_cat_ohe.shape, y_train.shape)
print("Shape of X_cv after one hot encodig ",X_cv_cat_ohe.shape, y_cv.shape)
print("Shape of X_test after one hot encodig ",X_test_cat_ohe.shape, y_test.shape)
```

```
After Vectorizations
Shape of X_train after one hot encodig  (49041, 9) (49041,)
Shape of X_cv after one hot encodig  (24155, 9) (24155,)
Shape of X_test after one hot encodig  (36052, 9) (36052,)
```

## Subcategories

In [31]:

```
vec2 = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), binary=True)
X_train_sub_cat_ohe = vec2.fit_transform(X_train['clean_subcategories'].values)
X_cv_sub_cat_ohe = vec2.transform(X_cv['clean_subcategories'].values)
X_test_sub_cat_ohe = vec2.transform(X_test['clean_subcategories'].values)

print("After Vectorizations")
print("Shape of X_train after one hot encodig ",X_train_sub_cat_ohe.shape, y_train.shap
e)
print("Shape of X_cv after one hot encodig ",X_cv_sub_cat_ohe.shape, y_cv.shape)
print("Shape of X_test after one hot encodig ",X_test_sub_cat_ohe.shape, y_test.shape)
```

```
After Vectorizations
Shape of X_train after one hot encodig  (49041, 30) (49041,)
Shape of X_cv after one hot encodig  (24155, 30) (24155,)
Shape of X_test after one hot encodig  (36052, 30) (36052,)
```

## School state

In [32]:

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
school_dict = dict(my_counter)
sorted_school_dict = dict(sorted(school_dict.items(), key=lambda kv: kv[1]))
```

In [33]:

```
vec3 = CountVectorizer(vocabulary=list(sorted_school_dict.keys()), binary=True)
X_train_state_ohe = vec3.fit_transform(X_train['school_state'].values)
X_cv_state_ohe = vec3.transform(X_cv['school_state'].values)
X_test_state_ohe = vec3.transform(X_test['school_state'].values)

print("After vectorizations")
print("Shape of X_train after one hot encodig ",X_train_state_ohe.shape, y_train.shape)
print("Shape of X_cv after one hot encodig ",X_cv_state_ohe.shape, y_cv.shape)
print("Shape of X_test after one hot encodig ",X_test_state_ohe.shape, y_test.shape)
```

```
After vectorizations
Shape of X_train after one hot encodig  (49041, 51) (49041,)
Shape of X_cv after one hot encodig  (24155, 51) (24155,)
Shape of X_test after one hot encodig  (36052, 51) (36052,)
```

## teacher prefix

In [34]:

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
teacher_dict = dict(my_counter)
sorted_teacher_dict = dict(sorted(teacher_dict.items(), key=lambda kv: kv[1]))
```

In [35]:

```
vec4 = CountVectorizer(vocabulary=list(sorted_teacher_dict.keys()), binary=True)
X_train_teacher_ohe = vec4.fit_transform(X_train['teacher_prefix'].values) # fit has to
happen only on train data
X_cv_teacher_ohe = vec4.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe = vec4.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print("Shape of X_train after one hot encodig ",X_train_teacher_ohe.shape, y_train.shap
e)
print("Shape of X_cv after one hot encodig ",X_cv_teacher_ohe.shape, y_cv.shape)
print("Shape of X_test after one hot encodig ",X_test_teacher_ohe.shape, y_test.shape)
```

```
After vectorizations
Shape of X_train after one hot encodig  (49041, 5) (49041,)
Shape of X_cv after one hot encodig  (24155, 5) (24155,)
Shape of X_test after one hot encodig  (36052, 5) (36052,)
```

## Project grade category

In [36]:

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
grade_dict = dict(my_counter)
sorted_grade_dict = dict(sorted(grade_dict.items(), key=lambda kv: kv[1]))
```

In [37]:

```
vec5 = CountVectorizer(vocabulary=list(sorted_grade_dict.keys()), binary=True)
X_train_grade_ohe = vec5.fit_transform(X_train['project_grade_category'].values) # fit
 has to happen only on train data
X_cv_grade_ohe = vec5.transform(X_cv['project_grade_category'].values)
X_test_grade_ohe = vec5.transform(X_test['project_grade_category'].values)
```

In [38]:

```
print('After Vectorizations')
print("Shape of X_train after one hot encodig ",X_train_grade_ohe.shape, y_train.shape)
print("Shape of X_cv after one hot encodig ",X_cv_grade_ohe.shape, y_cv.shape)
print("Shape of X_test after one hot encodig ",X_test_grade_ohe.shape, y_test.shape)
```

```
After Vectorizations
Shape of X_train after one hot encodig  (49041, 5) (49041,)
Shape of X_cv after one hot encodig  (24155, 5) (24155,)
Shape of X_test after one hot encodig  (36052, 5) (36052,)
```

## Bag of words on essay

In [39]:

```
vec6 = CountVectorizer(min_df=10)

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vec6.fit_transform(preprocessed_train_essay)# fit has to happen onl
y on train data
X_cv_essay_bow = vec6.transform(preprocessed_cv_essay)
X_test_essay_bow = vec6.transform(preprocessed_test_essay)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
```

```
After vectorizations
(49041, 12032) (49041,)
(24155, 12032) (24155,)
(36052, 12032) (36052,)
```

## Bag of words on title

In [40]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vec7 = CountVectorizer(min_df=10)

# fit has to happen only on train data
# we use the fitted CountVectorizer to convert the text to vector
X_train_title_bow = vec7.fit_transform(preprocessed_train_title)
X_cv_title_bow = vec7.transform(preprocessed_cv_title)
X_test_title_bow = vec7.transform(preprocessed_test_title)

print("After vectorizations")
print(X_train_title_bow.shape, y_train.shape)
print(X_cv_title_bow.shape, y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
```

```
After vectorizations
(49041, 1989) (49041,)
(24155, 1989) (24155,)
(36052, 1989) (36052,)
```

## tfidf on essay

In [41]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vec8 = TfidfVectorizer(min_df=10)

# fit has to happen only on train data
X_train_ess_tfidf = vec8.fit_transform(preprocessed_train_essay)

# we use the fitted Tfidf Vectorizer to convert the text to vector
X_cv_ess_tfidf = vec8.transform(preprocessed_cv_essay)
X_test_ess_tfidf = vec8.transform(preprocessed_test_essay)

print("After vectorizations")
print(X_train_ess_tfidf.shape, y_train.shape)
print(X_cv_ess_tfidf.shape, y_cv.shape)
print(X_test_ess_tfidf.shape, y_test.shape)
```

```
After vectorizations
(49041, 12032) (49041,)
(24155, 12032) (24155,)
(36052, 12032) (36052,)
```

## tfidf on title

In [42]:

```
vec9 = TfidfVectorizer(min_df=10)
X_train_title_tfidf = vec9.fit_transform(preprocessed_train_title)

# we use the fitted Tfidf Vectorizer to convert the text to vector
X_cv_title_tfidf = vec9.transform(preprocessed_cv_title)
X_test_title_tfidf = vec9.transform(preprocessed_test_title)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
```

```
After vectorizations
(49041, 1989) (49041,)
(24155, 1989) (24155,)
(36052, 1989) (36052,)
```

## price

In [43]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-al
l-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_i
ndex()
price_data.head(2)
```

Out[43]:

| | id | price | quantity |
|---|---|---|---|
| **0** | p000001 | 459.56 | 7 |
| **1** | p000002 | 515.89 | 21 |

In [44]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_i
ndex()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [45]:

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

In [46]:

```
project_data.isnull().any()
```

Out[46]:

```
Unnamed: 0                                      False
id                                              False
teacher_id                                      False
teacher_prefix                                  False
school_state                                    False
project_submitted_datetime                      False
project_grade_category                          False
project_title                                   False
project_essay_1                                 False
project_essay_2                                 False
project_essay_3                                  True
project_essay_4                                  True
project_resource_summary                        False
teacher_number_of_previously_posted_projects    False
project_is_approved                             False
essay                                           False
clean_categories                                False
clean_subcategories                             False
price_x                                         False
quantity_x                                      False
price_y                                         False
quantity_y                                      False
dtype: bool
```

In [47]:

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

In [48]:

```
# https://stackoverflow.com/questions/32617811/imputation-of-missing-values-for-categor
ies-in-pandas
#replacing nan with most frequently occuring element
X_train['price'] = X_train['price'].fillna(X_train['price'].mode().iloc[0])
X_cv['price'] = X_cv['price'].fillna(X_train['price'].mode().iloc[0])
X_test['price'] = X_test['price'].fillna(X_test['price'].mode().iloc[0])
print(X_train['price'].isnull().sum())
print(X_cv['price'].isnull().sum())
print(X_test['price'].isnull().sum())
```

```
0
0
0
```

## Standardizing price, quantity, previous projects

In [49]:

```
# price
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.pr
eprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ...
 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = Normalizer()

X_train_price_stndrd = price_scalar.fit_transform(X_train['price'].values.reshape(1,-1
))
X_cv_price_stndrd = price_scalar.transform(X_cv['price'].values.reshape(1,-1))
X_test_price_stndrd = price_scalar.transform(X_test['price'].values.reshape(1,-1))
```

In [50]:

```
X_train_price_stndrd = X_train_price_stndrd.reshape(-1,1)
X_cv_price_stndrd = X_cv_price_stndrd.reshape(-1,1)
X_test_price_stndrd = X_test_price_stndrd.reshape(-1,1)
```

In [51]:

```
X_train_price_stndrd
```

Out[51]:

```
array([[0.00249893],
       [0.0017855 ],
       [0.00484005],
       ...,
       [0.00242904],
       [0.00393886],
       [0.00064567]])
```

In [52]:

```
X_cv_price_stndrd
```

Out[52]:

```
array([[0.01528047],
       [0.00348139],
       [0.00383149],
       ...,
       [0.01305852],
       [0.00431142],
       [0.00475329]])
```

In [53]:

```
X_test_price_stndrd
```

Out[53]:

```
array([[0.01102928],
       [0.00106705],
       [0.00044305],
       ...,
       [0.00012185],
       [0.00177986],
       [0.00309936]])
```

## quantity

In [54]:

```
# https://stackoverflow.com/questions/32617811/imputation-of-missing-values-for-categor
ies-in-pandas
#replacing nan with most frequently occuring element

X_train['quantity'] = X_train['quantity'].fillna(X_train['quantity'].mode().iloc[0])
X_cv['quantity'] = X_cv['quantity'].fillna(X_cv['quantity'].mode().iloc[0])
X_test['quantity'] = X_test['quantity'].fillna(X_test['quantity'].mode().iloc[0])
```

In [55]:

```
# quantity
quantity_scalar = Normalizer()
X_train_quantity_stndrd = quantity_scalar.fit_transform(X_train['quantity'].values.resh
ape(1,-1))
X_cv_quantity_stndrd = quantity_scalar.transform(X_cv['quantity'].values.reshape(1,-1))
X_test_quantity_stndrd = quantity_scalar.transform(X_test['quantity'].values.reshape(1,
-1))
```

In [56]:

```
X_train_quantity_stndrd = X_train_quantity_stndrd.reshape(-1,1)
X_cv_quantity_stndrd = X_cv_quantity_stndrd.reshape(-1,1)
X_test_quantity_stndrd = X_test_quantity_stndrd.reshape(-1,1)
```

In [57]:

```
X_train_quantity_stndrd
```

Out[57]:

```
array([[0.00590961],
       [0.00265933],
       [0.00132966],
       ...,
       [0.0014774 ],
       [0.00295481],
       [0.00206836]])
```

In [58]:

```
X_cv_quantity_stndrd
```

Out[58]:

```
array([[0.00145398],
       [0.00706219],
       [0.00249254],
       ...,
       [0.00166169],
       [0.00124627],
       [0.00020771]])
```

In [59]:

```
X_test_quantity_stndrd
```

Out[59]:

```
array([[0.00016363],
       [0.00032727],
       [0.00196362],
       ...,
       [0.00490904],
       [0.0042545 ],
       [0.00539995]])
```

In [60]:

```
# https://stackoverflow.com/questions/32617811/imputation-of-missing-values-for-categor
ies-in-pandas
#replacing nan with most frequently occuring element

X_train['teacher_number_of_previously_posted_projects'] = X_train['teacher_number_of_pr
eviously_posted_projects'].fillna(X_train['teacher_number_of_previously_posted_project
s'].mode().iloc[0])
X_cv['teacher_number_of_previously_posted_projects'] = X_cv['teacher_number_of_previous
ly_posted_projects'].fillna(X_cv['teacher_number_of_previously_posted_projects'].mode()
.iloc[0])
X_test['teacher_number_of_previously_posted_projects'] = X_test['teacher_number_of_prev
iously_posted_projects'].fillna(X_test['teacher_number_of_previously_posted_projects'].
mode().iloc[0])
```

## previous projects

In [61]:

```
# previous_year_projects
price_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape
(-1,1)) # finding the mean and standard deviation of this data
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var
_[0])}")
# Now standardize the data with above mean and variance.
X_train_prev_proj_stndrd =price_scalar.transform(X_train['teacher_number_of_previously_
posted_projects'].values.reshape(1,-1))
# Now standardize the data with above mean and variance.
X_test_prev_proj_stndrd =price_scalar.transform(X_test['teacher_number_of_previously_po
sted_projects'].values.reshape(1,-1))
# Now standardize the data with above mean and variance.
X_cv_prev_proj_stndrd = price_scalar.transform(X_cv['teacher_number_of_previously_poste
d_projects'].values.reshape(1,-1))
```

In [62]:

```
X_train_prev_proj_stndrd = X_train_prev_proj_stndrd.reshape(-1,1)
X_cv_prev_proj_stndrd = X_cv_prev_proj_stndrd.reshape(-1,1)
X_test_prev_proj_stndrd = X_test_prev_proj_stndrd.reshape(-1,1)
```

In [63]:

```
X_train_prev_proj_stndrd
```

Out[63]:

```
array([[0.00091065],
       [0.01487387],
       [0.00121419],
       ...,
       [0.00075887],
       [0.        ],
       [0.00015177]])
```

In [64]:

```
X_cv_prev_proj_stndrd
```

Out[64]:

```
array([[0.        ],
       [0.        ],
       [0.00064264],
       ...,
       [0.00085686],
       [0.00128529],
       [0.00064264]])
```

In [65]:

```
X_test_prev_proj_stndrd
```

Out[65]:

```
array([[0.00087467],
       [0.0005248 ],
       [0.00332374],
       ...,
       [0.        ],
       [0.00017493],
       [0.00122454]])
```

In [66]:

```
X_train.head()
```

Out[66]:

| | id | teacher_prefix | school_state | project_grade_category | project_title | projec |
|---|---|---|---|---|---|---|
| **0** | p032563 | Mrs. | AR | Grades 9-12 | Modeling Minds | In our school studer |
| **1** | p146236 | Mr. | NY | Grades PreK-2 | Adding to Our TODDally PARRfect Library! | Do you remen first bo read tl |
| **2** | p126691 | Mrs. | CT | Grades PreK-2 | Tracking Our Way to a Healthy Target | My stu active, and ea |
| **3** | p115815 | Mrs. | FL | Grades PreK-2 | You Read to Me and I'll Read to You! | \"Tell r forget. and I r |
| **4** | p180285 | Mrs. | OK | Grades 3-5 | Protect Our Investment: Covers for Our Ipads | Hoora cart is you re |

In [67]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [68]:

```
print(X_train_cat_ohe.shape)
print(X_train_sub_cat_ohe.shape)
print(X_train_state_ohe.shape)
print(X_train_teacher_ohe.shape)
print(X_train_grade_ohe.shape)
print(X_train_essay_bow.shape)
print(X_train_ess_tfidf.shape)
print(X_train_title_bow.shape)
print(X_train_title_tfidf.shape)
print(X_train_price_stndrd.shape)
print(X_train_quantity_stndrd.shape)
print(X_train_prev_proj_stndrd.shape)
```

```
(49041, 9)
(49041, 30)
(49041, 51)
(49041, 5)
(49041, 5)
(49041, 12032)
(49041, 12032)
(49041, 1989)
(49041, 1989)
(49041, 1)
(49041, 1)
(49041, 1)
```

## Concatenating all the features

In [69]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
X_train_bow = hstack((X_train_title_bow, X_train_essay_bow, X_train_teacher_ohe, X_trai
n_cat_ohe, X_train_sub_cat_ohe,
                      X_train_grade_ohe, X_train_state_ohe, X_train_price_stndrd, X_tra
in_quantity_stndrd, X_train_prev_proj_stndrd)).tocsr().toarray()
```

In [70]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
X_cv_bow = hstack((X_cv_title_bow,X_cv_essay_bow,X_cv_teacher_ohe,X_cv_cat_ohe,X_cv_sub
_cat_ohe,
                      X_cv_grade_ohe,X_cv_state_ohe,X_cv_price_stndrd, X_cv_quantity_st
ndrd, X_cv_prev_proj_stndrd)).tocsr().toarray()
```

In [71]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
X_test_bow = hstack((X_test_title_bow,X_test_essay_bow,X_test_teacher_ohe,X_test_cat_oh
e,X_test_sub_cat_ohe,
                      X_test_grade_ohe,X_test_state_ohe,X_test_price_stndrd, X_test_qua
ntity_stndrd, X_test_prev_proj_stndrd)).tocsr().toarray()
```

In [ ]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_bow = scaler.fit_transform(X_train_bow)
X_cv_bow = scaler.transform(X_cv_bow)
X_test_bow = scaler.transform(X_test_bow)
```

In [72]:

```
print("Final Data matrix")
print(X_train_bow.shape, y_train.shape)
print(X_cv_bow.shape, y_cv.shape)
print(X_test_bow.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 14124) (49041,)
(24155, 14124) (24155,)
(36052, 14124) (36052,)
==========================================================================
==========================
```

In [73]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
X_train_tfidf = hstack((X_train_title_tfidf,X_train_ess_tfidf,X_train_teacher_ohe,X_tra
in_cat_ohe,X_train_sub_cat_ohe,
                      X_train_grade_ohe,X_train_state_ohe,X_train_price_stndrd, X_train
_quantity_stndrd, X_train_prev_proj_stndrd)).tocsr().toarray()
```

In [74]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
X_cv_tfidf = hstack((X_cv_title_tfidf,X_cv_ess_tfidf,X_cv_teacher_ohe,X_cv_cat_ohe,X_cv
_sub_cat_ohe,
                      X_cv_grade_ohe,X_cv_state_ohe,X_cv_price_stndrd, X_cv_quantity_st
ndrd, X_cv_prev_proj_stndrd)).tocsr().toarray()
```

In [75]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
X_test_tfidf = hstack((X_test_title_tfidf,X_test_ess_tfidf,X_test_teacher_ohe,X_test_ca
t_ohe,X_test_sub_cat_ohe,
                      X_test_grade_ohe,X_test_state_ohe,X_test_price_stndrd, X_test_qua
ntity_stndrd, X_test_prev_proj_stndrd)).tocsr().toarray()
```

In [76]:

```
print('Final data matrix')
print(X_train_tfidf.shape, y_train.shape)
print(X_cv_tfidf.shape, y_cv.shape)
print(X_test_tfidf.shape, y_test.shape)
```

```
Final data matrix
(49041, 14124) (49041,)
(24155, 14124) (24155,)
(36052, 14124) (36052,)
```

## Naive Bayes on BOW featurization

In [77]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
 of the positive class
    # 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 =
 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [78]:

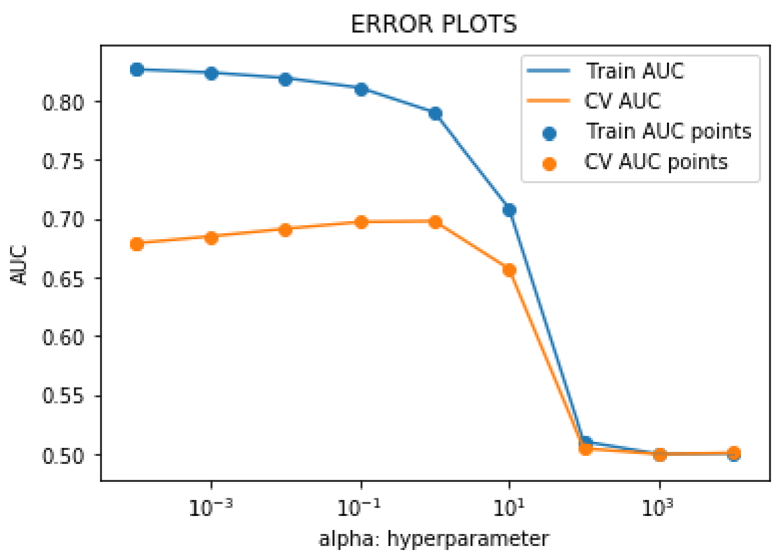
```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math

train_auc = []
cv_auc = []
alpha =[0.0001,0.0001,0.001,0.01,0.1,1,10,100,1000,10000]


for i  in   tqdm(alpha):

    neigh = MultinomialNB(alpha=i, class_prior = [0.5,0.5])# takes the alpha from the i
th list value
    neigh.fit(X_train_bow, y_train)# fit the model
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
he positive class
# not the predicted outputs
    y_train_pred=batch_predict(neigh,X_train_bow)
    y_cv_pred=batch_predict(neigh,X_cv_bow)
# roc curve
#Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from predicti
on scores.
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
plt.plot(alpha, train_auc, label='Train AUC')
plt.plot(alpha, cv_auc, label='CV AUC')
plt.xscale('log')# we take the log in the x axis
plt.scatter(alpha, train_auc, label='Train AUC points')
plt.scatter(alpha, cv_auc, label='CV AUC points')
plt.xscale('log')
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|██████████████████████████████████████████████████████
██████████| 10/10 [02:02<00:00, 12.21s/it]
```



## best alpha

In [79]:

```
optimal_alpha= alpha[cv_auc.index(max(cv_auc))]
alpha_values=[math.log(x) for x in alpha]
print('optimal alpha for which auc is maximum : ',optimal_alpha)
```

```
optimal alpha for which auc is maximum :  1
```

## Hyperparameter tuning

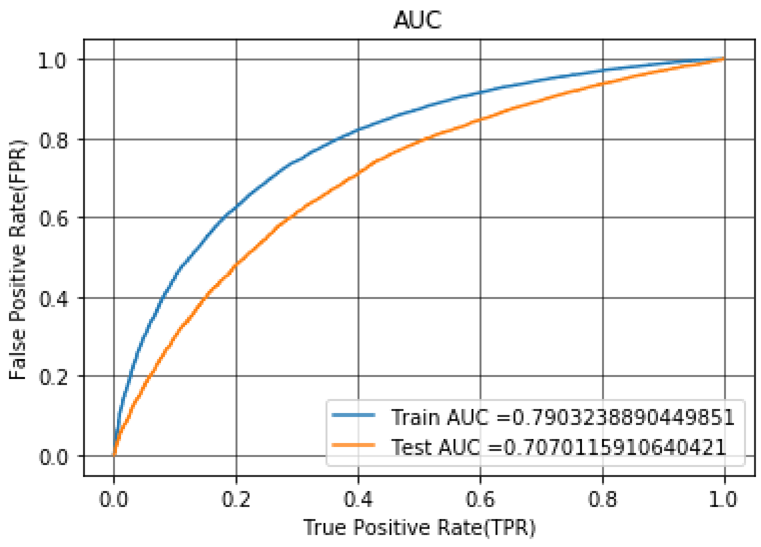
In [80]:

```
nb = MultinomialNB(alpha=optimal_alpha)
nb.fit(X_train_bow, y_train)

y_train_pred = batch_predict(nb, X_train_bow)
y_test_pred = batch_predict(nb, X_test_bow)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



## best threshold

In [81]:

```
# 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(threshold, fpr, tpr):
    t = threshold[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.rou
nd(t,3))
    return t

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

## Confusion matrix

In [82]:

```
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)))
print('='*100)
```

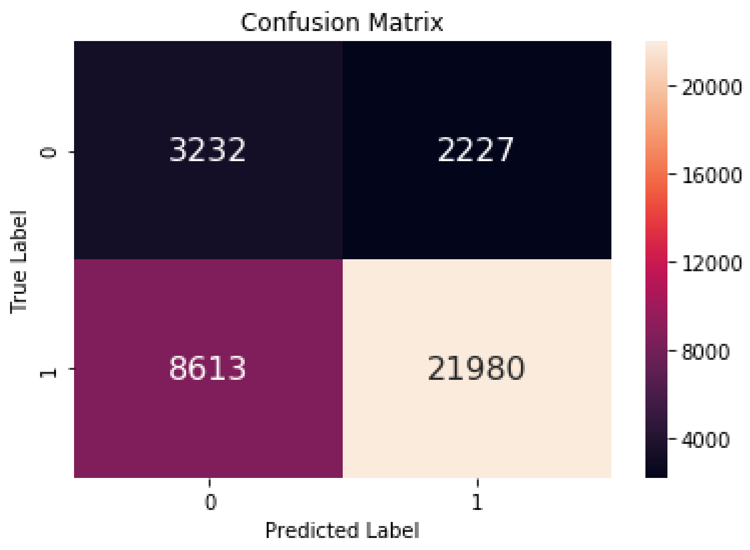
```
the maximum value of tpr*(1-fpr) 0.5223255128845629 for threshold 0.798
Train confusion matrix
[[ 5269  2157]
 [10980 30635]]
Test confusion matrix
[[ 3232  2227]
 [ 8613 21980]]
===========================================================================
=========================
```

In [83]:

```
#stackoverflow.com/questions/54018742/valueerror-classification-metrics-cant-handle-a-m
ix-of-unknown-and-binary-targ
matrix = confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
sns.heatmap(matrix, annot=True, annot_kws={'size':16}, fmt='g')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
```

Out[83]:

```
Text(0.5, 1, 'Confusion Matrix')
```

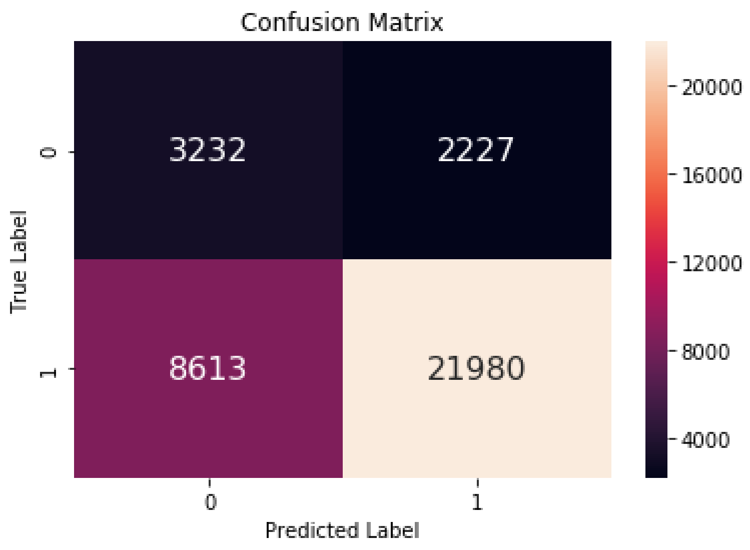


In [84]:

```
#stackoverflow.com/questions/54018742/valueerror-classification-metrics-cant-handle-a-m
ix-of-unknown-and-binary-targ
matrix = confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
sns.heatmap(matrix, annot=True, annot_kws={'size':16}, fmt='g')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
```

Out[84]:

```
Text(0.5, 1, 'Confusion Matrix')
```



## Naive Bayes on tfidf featurization

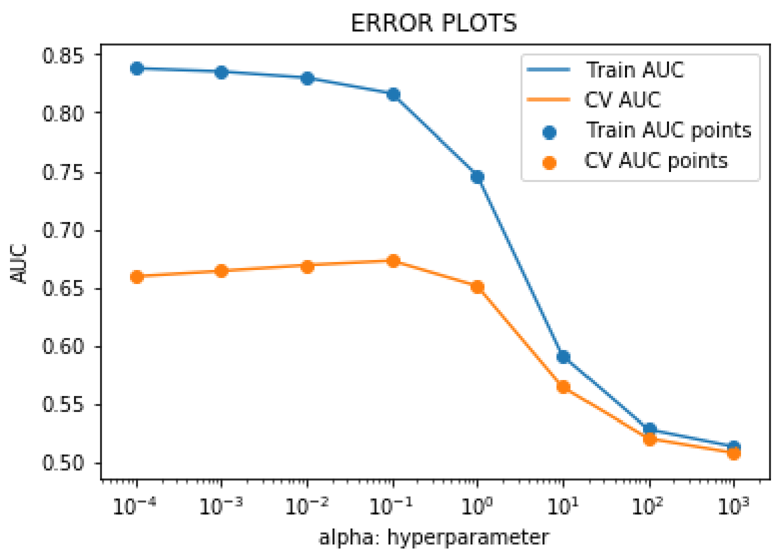
In [85]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
import math
train_auc = []
cv_auc = []
alpha =[0.0001,0.001,0.01,0.1,1,10,100,1000]


for i  in   tqdm(alpha):

    neigh = MultinomialNB(alpha=i)# takes the alpha from the i th list value
    neigh.fit(X_train_tfidf, y_train)# fit the model
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
he positive class
# not the predicted outputs
    y_train_pred=batch_predict(neigh,X_train_tfidf)
    y_cv_pred=batch_predict(neigh,X_cv_tfidf)
# roc curve
#Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC) from predicti
on scores.
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
plt.plot(alpha, train_auc, label='Train AUC')
plt.plot(alpha, cv_auc, label='CV AUC')
plt.xscale('log')# we take the log in the x axis
plt.scatter(alpha, train_auc, label='Train AUC points')
plt.scatter(alpha, cv_auc, label='CV AUC points')
plt.xscale('log')
plt.legend()
plt.xlabel("alpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.show()
```

```
100%|████████████████████████████████████████████████████████████
█████████████| 8/8 [02:03<00:00, 15.43s/it]
```



## best alpha

In [86]:

```
optimal_alpha= alpha[cv_auc.index(max(cv_auc))]
alpha_values=[math.log(x) for x in alpha]
print('optimal alpha for which auc is maximum : ',optimal_alpha)
```

```
optimal alpha for which auc is maximum :  0.1
```

## Hyperparameter tuning

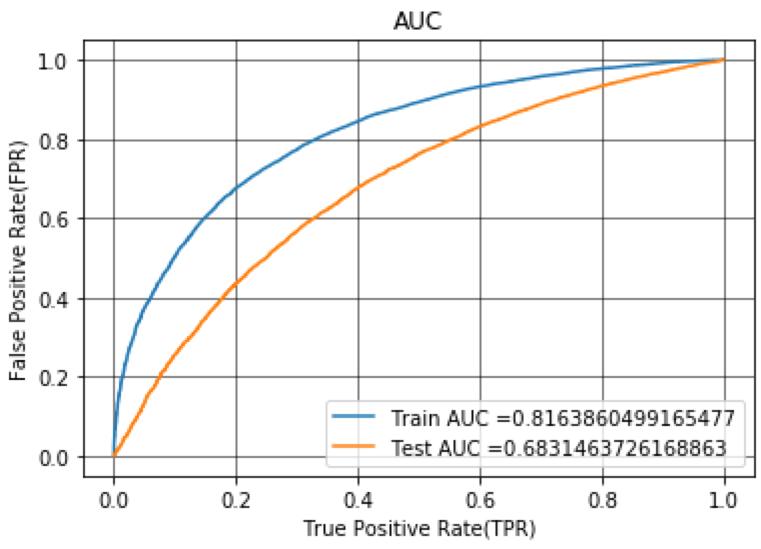
In [87]:

```
nb = MultinomialNB(alpha=optimal_alpha)
nb.fit(X_train_tfidf, y_train)

y_train_pred = batch_predict(nb, X_train_tfidf)
y_test_pred = batch_predict(nb, X_test_tfidf)


train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid(color='black', linestyle='-', linewidth=0.5)
plt.show()
```



## Best threshold

In [88]:

```
# 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(threshold, fpr, tpr):
    t = threshold[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.rou
nd(t,3))
    return t

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

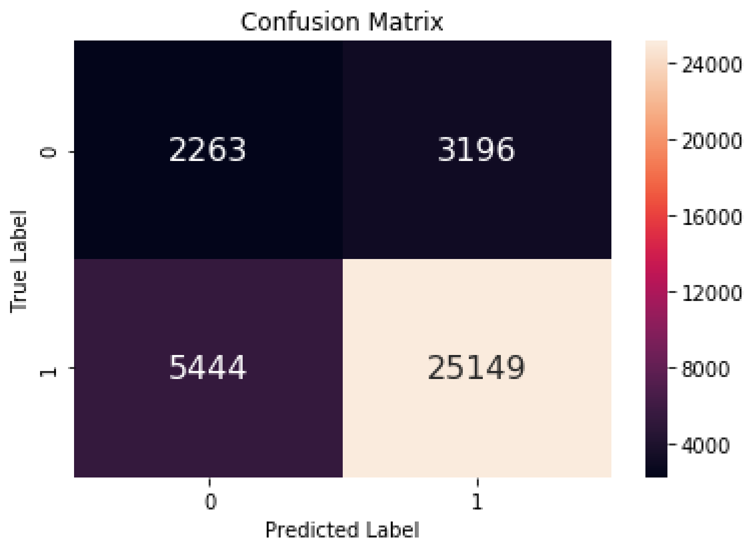
## Confusion Matrix

In [89]:

```
#stackoverflow.com/questions/54018742/valueerror-classification-metrics-cant-handle-a-m
ix-of-unknown-and-binary-targ
matrix = confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
sns.heatmap(matrix, annot=True, annot_kws={'size':16}, fmt='g')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
```

Out[89]:

```
Text(0.5, 1, 'Confusion Matrix')
```

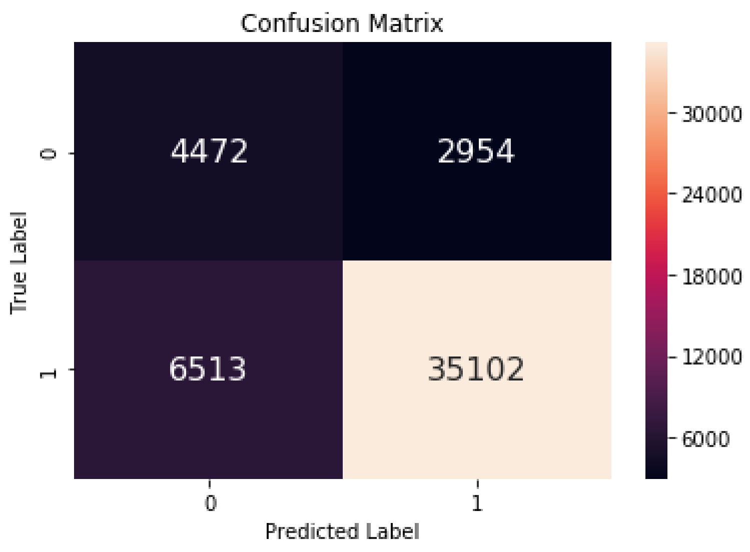


In [90]:

```
#stackoverflow.com/questions/54018742/valueerror-classification-metrics-cant-handle-a-m
ix-of-unknown-and-binary-targ
matrix = confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t))
sns.heatmap(matrix, annot=True, annot_kws={'size':16}, fmt='g')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.title('Confusion Matrix')
```

Out[90]:

```
Text(0.5, 1, 'Confusion Matrix')
```



## top 20 postive features

### for bow

In [91]:

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

Out[91]:

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

In [96]:

```
bow_features_names = []

for a in vec1.get_feature_names() :
    bow_features_names.append(a)

for a in vec2.get_feature_names() :
    bow_features_names.append(a)

for a in vec3.get_feature_names() :
    bow_features_names.append(a)

for a in vec4.get_feature_names() :
    bow_features_names.append(a)

for a in vec5.get_feature_names() :
    bow_features_names.append(a)

bow_features_names.append("price")
bow_features_names.append("quantity")
bow_features_names.append("teacher_number_of_previously_posted")

for a in vec6.get_feature_names() :
    bow_features_names.append(a)

for a in vec7.get_feature_names() :
    bow_features_names.append(a)

len(bow_features_names)
```

Out[96]:

```
14124
```

In [98]:

```
bow_features_probs1 = []
for a in range(len(bow_features_names)) :
    b = nb_bow.feature_log_prob_[1,a]
    bow_features_probs1.append(b)

len(bow_features_probs1)
```

Out[98]:

```
14124
```

In [99]:

```
final_features_bow_df_pos = pd.DataFrame({'feature_prob_estimates' : bow_features_probs
1,'feature_names' : bow_features_names})
final_features_bow_df_pos.sort_values(by = ['feature_prob_estimates'], ascending = Fals
e,inplace=True)
final_features_bow_df_pos.head(20)
```

Out[99]:

| | feature_prob_estimates | feature_names |
|---|---|---|
| **12374** | -2.955196 | calculating |
| **11398** | -4.100399 | understand |
| **8173** | -4.465084 | portion |
| **3989** | -4.493199 | even |
| **9208** | -4.754470 | revive |
| **8169** | -4.798259 | portable |
| **7083** | -4.829524 | movers |
| **8540** | -4.972324 | proximal |
| **9040** | -4.989906 | replay |
| **9089** | -5.097971 | reservation |
| **13894** | -5.098671 | talented |
| **10640** | -5.122768 | surroundings |
| **13424** | -5.157362 | owls |
| **8405** | -5.251187 | producing |
| **4773** | -5.280015 | generational |
| **2198** | -5.289869 | collaborators |
| **4126** | -5.321531 | experiments |
| **3975** | -5.324440 | ethnic |
| **13935** | -5.369865 | thing |
| **12693** | -5.411498 | elementary |

## top 20 negative features

### for bow

In [101]:

```
bow_features_probs2 = []
for a in range(len(bow_features_names)) :
    b = nb_bow.feature_log_prob_[0,a]
    bow_features_probs2.append(b)

len(bow_features_probs2)
```

Out[101]:

```
14124
```

In [102]:

```
final_features_bow_df_neg = pd.DataFrame({'feature_prob_estimates' : bow_features_probs
2,'feature_names' : bow_features_names})
final_features_bow_df_neg.sort_values(by = ['feature_prob_estimates'], ascending = Fals
e,inplace=True)
final_features_bow_df_neg.head(20)
```

Out[102]:

| | feature_prob_estimates | feature_names |
|---|---|---|
| **12374** | -2.971448 | calculating |
| **11398** | -4.063684 | understand |
| **8173** | -4.381474 | portion |
| **3989** | -4.542996 | even |
| **9208** | -4.728287 | revive |
| **8169** | -4.740681 | portable |
| **7083** | -4.774262 | movers |
| **9040** | -4.939147 | replay |
| **8540** | -4.981428 | proximal |
| **9089** | -5.074662 | reservation |
| **13894** | -5.092230 | talented |
| **4126** | -5.242845 | experiments |
| **8405** | -5.330733 | producing |
| **10640** | -5.336583 | surroundings |
| **11787** | -5.346480 | warmth |
| **4773** | -5.374811 | generational |
| **13424** | -5.374811 | owls |
| **8598** | -5.375900 | purposes |
| **2198** | -5.382898 | collaborators |
| **13971** | -5.410264 | track |

# top 20 postive features

## for tfidf

In [103]:

```
from sklearn.naive_bayes import MultinomialNB


nb_tfidf = MultinomialNB(alpha = 0.5,class_prior=[0.5,0.5])
nb_tfidf.fit(X_train_tfidf, y_train)
```

Out[103]:

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

In [104]:

```
tfidf_features_names = []

for a in vec1.get_feature_names() :
    tfidf_features_names.append(a)

for a in vec2.get_feature_names() :
    tfidf_features_names.append(a)

for a in vec3.get_feature_names() :
    tfidf_features_names.append(a)

for a in vec4.get_feature_names() :
    tfidf_features_names.append(a)

for a in vec5.get_feature_names() :
    tfidf_features_names.append(a)

tfidf_features_names.append("price")
tfidf_features_names.append("quantity")
tfidf_features_names.append("teacher_number_of_previously_posted")

for a in vec6.get_feature_names() :
    tfidf_features_names.append(a)

for a in vec7.get_feature_names() :
    tfidf_features_names.append(a)

len(tfidf_features_names)
```

Out[104]:

```
14124
```

In [105]:

```
tfidf_features_probs1 = []
for a in range(len(tfidf_features_names)) :
    b = nb_tfidf.feature_log_prob_[1,a]
    tfidf_features_probs1.append(b)

len(tfidf_features_probs1)
```

Out[105]:

```
14124
```

In [106]:

```
final_features_tfidf_df_pos = pd.DataFrame({'feature_prob_estimates' : tfidf_features_p
robs1,'feature_names' : tfidf_features_names})
final_features_tfidf_df_pos.sort_values(by = ['feature_prob_estimates'], ascending = Fa
lse,inplace=True)
final_features_tfidf_df_pos.head(20)
```

Out[106]:

| | feature_prob_estimates | feature_names |
|---|---|---|
| **12374** | -4.370389 | calculating |
| **11398** | -5.375919 | understand |
| **8173** | -5.558849 | portion |
| **3989** | -5.559836 | even |
| **10640** | -5.774190 | surroundings |
| **9208** | -5.811163 | revive |
| **8169** | -5.817428 | portable |
| **7083** | -5.821158 | movers |
| **1035** | -5.852889 | audio |
| **3323** | -5.924908 | disruptive |
| **8540** | -5.929365 | proximal |
| **12693** | -5.959678 | elementary |
| **13894** | -5.970765 | talented |
| **9089** | -6.000270 | reservation |
| **13424** | -6.006579 | owls |
| **8405** | -6.026945 | producing |
| **4773** | -6.068038 | generational |
| **13935** | -6.088671 | thing |
| **2198** | -6.088948 | collaborators |
| **3975** | -6.121091 | ethnic |

# top 20 negative features

## for tfidf

In [107]:

```
tfidf_features_probs2 = []
for a in range(len(tfidf_features_names)) :
    b = nb_tfidf.feature_log_prob_[0,a]
    tfidf_features_probs2.append(b)

len(tfidf_features_probs2)
```

Out[107]:

```
14124
```

In [108]:

```
final_features_tfidf_df_neg = pd.DataFrame({'feature_prob_estimates' : tfidf_features_p
robs2,'feature_names' : tfidf_features_names})
final_features_tfidf_df_neg.sort_values(by = ['feature_prob_estimates'], ascending = Fa
lse,inplace=True)
final_features_tfidf_df_neg.head(20)
```

Out[108]:

| | feature_prob_estimates | feature_names |
|---|---|---|
| **12374** | -4.442101 | calculating |
| **11398** | -5.397401 | understand |
| **8173** | -5.526102 | portion |
| **3989** | -5.664093 | even |
| **8169** | -5.807738 | portable |
| **7083** | -5.809520 | movers |
| **1035** | -5.837170 | audio |
| **9208** | -5.838742 | revive |
| **8598** | -5.951420 | purposes |
| **8540** | -5.996227 | proximal |
| **13894** | -6.018830 | talented |
| **9089** | -6.022372 | reservation |
| **10640** | -6.047918 | surroundings |
| **11787** | -6.114489 | warmth |
| **4126** | -6.131609 | experiments |
| **13658** | -6.141821 | rug |
| **8405** | -6.154902 | producing |
| **12693** | -6.165063 | elementary |
| **13971** | -6.187461 | track |
| **4773** | -6.213545 | generational |

In [110]:

```
from prettytable import PrettyTable
ptable = PrettyTable()
ptable.title = 'Classification Report'

ptable.field_names = ["Vectorization", "Model", "alpha", "AUC"]

ptable.add_row(["BOW","Naive Bayes",1,70.70])
ptable.add_row(["tf-idf", "Naive Bayes",0.1,68.31])

print(ptable)
```

```
+---------------+-------------+-------+-------+
| Vectorization |    Model    | alpha |  AUC  |
+---------------+-------------+-------+-------+
|      BOW      | Naive Bayes |   1   |  70.7 |
|     tf-idf    | Naive Bayes |  0.1  | 68.31 |
+---------------+-------------+-------+-------+
```

# Summary:

### Initially after loading the dataset if any null values exists replace them with most occuring element then split the data into training, validation, testing data and preprocessed the data to avoid the leakage. Applied bag of words, tfidf for featurizing the data. After concatenating the data applied Naive Bayes classifier.

### Created list of feature names corresponding to original data using countvectorizer.