```
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sqlite3
import re,string,math,operator,os
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.feature_extraction.text import CountVectorizer,TfidfTransformer,TfidfVectoriz
from sklearn.metrics import accuracy_score,roc_auc_score, confusion_matrix,auc
#from sklearn.preprocessing import
from tqdm import tqdm
from gensim.models import Word2Vec, keyedvectors
import pickle
from chart_studio.plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()

from collections import Counter
print("Done")
```

```
Done
```

1.1 Reading data

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

url = 'https://drive.google.com/uc?id=1OcMV5zjAJI7OvNxxN4Ant52BDF3jrZOZ'
output = 'resources.csv'
gdown.download(url, output, quiet=False)
import gdown

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

```
Downloading...
From: https://drive.google.com/uc?id=1OcMV5zjAJI7OvNxxN4Ant52BDF3jrZOZ
To: /content/resources.csv
127MB [00:00, 158MB/s]
Downloading...
From: https://drive.google.com/uc?id=1JGtsNLea4Q2HZQIgBp3pRrOfRN80qIg0
To: /content/train_data.csv
201MB [00:02, 77.6MB/s]
"\nfor loading data in jupyter\n%%time\ntrain=pd.read_csv('train_data.csv')\nresource
```

```
train=pd.read_csv('train_data.csv')
resources=pd.read_csv('resources.csv')
```

```
train.shape, train.columns.values
```

```
((109248, 17),
 array(['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'], dtype=object))
```

```
resources.shape, resources.columns.values
```

```
((1541272, 4), array(['id', 'description', 'quantity', 'price'], dtype=object))
```

```
counts=train['project_is_approved'].value_counts()
```

```
train.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL

```
resources.head(2)
```

```
            id                                    description  quantity   price
```

1.2 preprocessing of project_subject_categories

```
train['date']=pd.to_datetime(train['project_submitted_datetime'].values)
train.drop('project_submitted_datetime',axis=1,inplace=True)
train.sort_values(by=['date'],inplace=True)
```

```
print(train['teacher_prefix'].isna().sum())
train['teacher_prefix']=train['teacher_prefix'].fillna('Mrs.') #imputing missing values wi
print(train['teacher_prefix'].isna().sum())
train['teacher_prefix']=train['teacher_prefix'].str.replace('.','')  # Removing (.) from p

#we can also define some function and replace the values and use it as project_data['teach
```

```
3
0
```

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

```
Mrs        57272
Ms         38955
Mr         10648
Teacher     2360
Dr            13
Name: teacher_prefix, dtype: int64
```

```
train['project_grade_category'].values[:10]
```

```
array(['Grades PreK-2', 'Grades 3-5', 'Grades PreK-2', 'Grades PreK-2',
       'Grades 3-5', 'Grades 3-5', 'Grades 3-5', 'Grades 3-5',
       'Grades PreK-2', 'Grades 3-5'], dtype=object)
```

```
train['project_grade_category']=train['project_grade_category'].str.replace(' ','')
train['project_grade_category']=train['project_grade_category'].str.replace('-','_')
train['project_grade_category']=train['project_grade_category'].str.replace('Grades','')
```

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

```
Literacy & Language                           23655
Math & Science                                17072
Literacy & Language, Math & Science           14636
Health & Sports                               10177
Music & The Arts                               5180
Special Needs                                  4226
Literacy & Language, Special Needs             3961
Applied Learning                               3771
Math & Science, Literacy & Language            2289
Applied Learning, Literacy & Language          2191
History & Civics                               1851
Math & Science, Special Needs                  1840
Literacy & Language, Music & The Arts          1757
Math & Science, Music & The Arts               1642
Applied Learning, Special Needs                1467
History & Civics, Literacy & Language          1421
Health & Sports, Special Needs                 1391
Warmth, Care & Hunger                          1309
Math & Science, Applied Learning               1220
Applied Learning, Math & Science               1052
Literacy & Language, History & Civics           809
Health & Sports, Literacy & Language            803
Applied Learning, Music & The Arts              758
Math & Science, History & Civics                652
Literacy & Language, Applied Learning           636
Applied Learning, Health & Sports               608
Math & Science, Health & Sports                 414
History & Civics, Math & Science                322
History & Civics, Music & The Arts              312
Special Needs, Music & The Arts                 302
Health & Sports, Math & Science                 271
History & Civics, Special Needs                 252
Health & Sports, Applied Learning               192
Applied Learning, History & Civics              178
Health & Sports, Music & The Arts               155
Music & The Arts, Special Needs                 138
Literacy & Language, Health & Sports             72
Health & Sports, History & Civics                43
Special Needs, Health & Sports                   42
History & Civics, Applied Learning               42
Health & Sports, Warmth, Care & Hunger           23
Special Needs, Warmth, Care & Hunger             23
Music & The Arts, Health & Sports                19
Music & The Arts, History & Civics               18
History & Civics, Health & Sports                13
Math & Science, Warmth, Care & Hunger            11
Applied Learning, Warmth, Care & Hunger          10
Music & The Arts, Applied Learning               10
Literacy & Language, Warmth, Care & Hunger        9
Music & The Arts, Warmth, Care & Hunger           2
History & Civics, Warmth, Care & Hunger           1
Name: project_subject_categories, dtype: int64
```

```
train['project_subject_categories'].values[:100]
```

```
array(['Math & Science', 'Special Needs', 'Literacy & Language',
       'Applied Learning', 'Literacy & Language',
       'Math & Science, History & Civics',
       'Literacy & Language, Math & Science',
       'Math & Science, History & Civics', 'Literacy & Language',
       'Math & Science', 'Literacy & Language',
       'Applied Learning, Music & The Arts',
       'Math & Science, Applied Learning',
       'Math & Science, Literacy & Language', 'Math & Science',
       'Literacy & Language, Math & Science',
       'History & Civics, Literacy & Language', 'Literacy & Language',
       'Literacy & Language', 'Literacy & Language',
       'Applied Learning, Health & Sports',
       'Math & Science, Music & The Arts', 'Literacy & Language',
       'Math & Science', 'Special Needs', 'Literacy & Language',
       'Literacy & Language', 'Literacy & Language',
       'Math & Science, Music & The Arts', 'Literacy & Language',
       'Applied Learning, Literacy & Language', 'Music & The Arts',
       'Health & Sports', 'Literacy & Language', 'Literacy & Language',
       'Math & Science', 'Music & The Arts', 'Literacy & Language',
       'Literacy & Language, Math & Science',
       'Applied Learning, Literacy & Language',
       'Math & Science, Applied Learning', 'Literacy & Language',
       'Applied Learning, Music & The Arts', 'Health & Sports',
       'Literacy & Language', 'Literacy & Language, Special Needs',
       'Math & Science, Special Needs',
       'Math & Science, Applied Learning',
       'Literacy & Language, Math & Science',
       'Math & Science, Literacy & Language',
       'Math & Science, Music & The Arts',
       'Literacy & Language, Math & Science', 'Math & Science',
       'Applied Learning', 'Literacy & Language',
       'Applied Learning, History & Civics', 'Literacy & Language',
       'Applied Learning', 'Math & Science',
       'Applied Learning, Special Needs', 'Literacy & Language',
       'Applied Learning', 'Literacy & Language, Math & Science',
       'Literacy & Language', 'Literacy & Language, Math & Science',
       'Health & Sports, Literacy & Language',
       'Literacy & Language, Special Needs', 'Math & Science',
       'Literacy & Language', 'Literacy & Language, Special Needs',
       'Literacy & Language', 'Literacy & Language, Music & The Arts',
       'Literacy & Language', 'Literacy & Language, Math & Science',
       'Math & Science, Music & The Arts', 'Health & Sports',
       'Math & Science', 'Health & Sports',
       'Applied Learning, Music & The Arts',
       'Math & Science, History & Civics',
       'Literacy & Language, Math & Science', 'Math & Science',
       'Literacy & Language, Math & Science', 'Literacy & Language',
       'Literacy & Language', 'Literacy & Language',
       'History & Civics, Math & Science', 'Literacy & Language',
       'Literacy & Language', 'Literacy & Language',
       'Literacy & Language', 'Health & Sports',
       'Literacy & Language, Math & Science',
       'Math & Science, History & Civics', 'Math & Science',
       'Applied Learning, Literacy & Language',
       'Literacy & Language, Special Needs', 'Health & Sports',
       'Math & Science, Literacy & Language', 'Literacy & Language'],
      dtype=object)
```

```
train['project_subject_categories']=train['project_subject_categories'].str.replace(" ",''
train['project_subject_categories']=train['project_subject_categories'].str.replace("&",'_
train['project_subject_categories']=train['project_subject_categories'].str.replace("The",
cat_list = []
for i in train['project_subject_categories'].values:
    temp = ""
    for j in i.split(','):
        temp+=j.strip()+" "
    cat_list.append(temp.strip().lower())

train['clean_categories'] = cat_list
train.drop('project_subject_categories', axis=1, inplace=True)
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in train['clean_categories'].values:
    my_counter.update(word.split())

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

```
The Worlds in sorted_cat_dict {'warmth': 1388, 'care_hunger': 1388, 'history_civics':
```

```
sorted_cat_dict
```

```
{'appliedlearning': 12135,
 'care_hunger': 1388,
 'health_sports': 14223,
 'history_civics': 5914,
 'literacy_language': 52239,
 'math_science': 41421,
 'music_arts': 10293,
 'specialneeds': 13642,
 'warmth': 1388}
```

```
train['project_subject_subcategories'].values[:100]
```

```
array(['Applied Sciences, Health & Life Science', 'Special Needs',
       'Literacy', 'Early Development', 'Literacy',
       'Mathematics, Social Sciences', 'Literacy, Mathematics',
       'Applied Sciences, History & Geography', 'ESL, Literacy',
       'Applied Sciences, Mathematics', 'Literacy',
       'Extracurricular, Visual Arts',
       'Applied Sciences, Early Development',
       'Environmental Science, Literacy',
       'Applied Sciences, Environmental Science', 'Literacy, Mathematics',
       'History & Geography, Literature & Writing', 'Literacy',
       'Literacy', 'Literacy, Literature & Writing',
       'Early Development, Gym & Fitness',
       'Environmental Science, Visual Arts',
       'Literacy, Literature & Writing',
       'Environmental Science, Mathematics', 'Special Needs',
       'ESL, Literacy', 'ESL, Literacy', 'ESL, Literacy',
       'Applied Sciences, Visual Arts', 'Literacy, Literature & Writing',
       'Early Development, Literacy', 'Music', 'Team Sports', 'Literacy',
       'Literacy', 'Health & Life Science, Mathematics',
       'Music, Performing Arts', 'Literacy, Literature & Writing',
       'ESL, Environmental Science', 'College & Career Prep, ESL',
       'Applied Sciences, Other', 'Literacy',
       'College & Career Prep, Visual Arts', 'Team Sports',
       'ESL, Literacy', 'Literature & Writing, Special Needs',
       'Health & Life Science, Special Needs', 'Applied Sciences, Other',
       'Literacy, Mathematics',
       'Environmental Science, Literature & Writing',
       'Applied Sciences, Visual Arts', 'Literacy, Mathematics',
       'Health & Life Science, Mathematics',
       'College & Career Prep, Other', 'Literacy, Literature & Writing',
       'Character Education, Social Sciences', 'Literature & Writing',
       'Early Development, Other',
       'Environmental Science, Health & Life Science',
       'Other, Special Needs', 'Foreign Languages',
       'College & Career Prep', 'Literacy, Mathematics', 'Literacy',
       'Literature & Writing, Mathematics',
       'Health & Wellness, Literature & Writing',
       'Literacy, Special Needs',
       'Environmental Science, Health & Life Science',
       'Literacy, Literature & Writing', 'Literacy, Special Needs',
       'Literature & Writing', 'Literacy, Visual Arts',
       'Literacy, Literature & Writing',
       'Literature & Writing, Mathematics',
       'Health & Life Science, Visual Arts', 'Gym & Fitness, Team Sports',
       'Mathematics', 'Health & Wellness, Team Sports',
       'College & Career Prep, Visual Arts',
       'Applied Sciences, Civics & Government', 'Literacy, Mathematics',
       'Applied Sciences, Environmental Science', 'Literacy, Mathematics',
       'Literacy', 'Literacy, Literature & Writing', 'Literacy',
       'Economics, Mathematics', 'Literacy', 'Literacy',
       'Literature & Writing', 'ESL, Literature & Writing', 'Team Sports',
       'Literacy, Mathematics', 'Environmental Science, Social Sciences',
       'Health & Life Science, Mathematics',
       'Early Development, Literacy', 'Literacy, Special Needs',
       'Health & Wellness', 'Health & Life Science, Literature & Writing',
       'Literacy'], dtype=object)
```

```
train['project_subject_subcategories']=train['project_subject_subcategories'].str.replace(
train['project_subject_subcategories']=train['project_subject_subcategories'].str.replace(
train['project_subject_subcategories']=train['project_subject_subcategories'].str.replace(
#train[''] = train['project_subject_subcategories']
#train.drop(['project_subject_subcategories'], axis=1, inplace=True)
cat_list = []
for i in train['project_subject_subcategories'].values:
    temp = ""
    for j in i.split(','):
        temp+=j.strip()+" "
    cat_list.append(temp.strip().lower())

train['clean_subcategories'] = cat_list
train.drop('project_subject_subcategories', axis=1, inplace=True)
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in train['clean_categories'].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]))
print("The Worlds in sorted_subcat_dict",sorted_sub_cat_dict)
```

```
The Worlds in sorted_subcat_dict {'warmth': 1388, 'care_hunger': 1388, 'history_civic
```

```
sorted_sub_cat_dict
```

```
{'appliedlearning': 12135,
 'care_hunger': 1388,
 'health_sports': 14223,
 'history_civics': 5914,
 'literacy_language': 52239,
 'math_science': 41421,
 'music_arts': 10293,
 'specialneeds': 13642,
 'warmth': 1388}
```

Adding a new feature Number of words in title

```
train['project_title'].str.len()
```

```
55660    44
76127    23
51140    46
473      38
41558    38
         ..
87154    35
14678    23
39096    74
87881    30
78306    13
Name: project_title, Length: 109248, dtype: int64
```

```
train["title_word_count"] = train['project_title'].str.split().str.len()
```

```
train.head()
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_st
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs	
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms	
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs	
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs	
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs	

combining 4 essays into 1 essay

```
train['essay']=train['project_essay_1'].map(str)+train['project_essay_2'].map(str)+train['
```

```
train['essay']
```

```
55660    I have been fortunate enough to use the Fairy ...
76127    Imagine being 8-9 years old. You're in your th...
51140    Having a class of 24 students comes with diver...
473      I recently read an article about giving studen...
41558    My students crave challenge, they eat obstacle...
                               ...                        
87154    Our day starts with about 100 students athlete...
14678    My students range from age four to five years ...
39096    We are a Title 1 school  650 total students. O...
87881    I teach many different types of students.  My ...
78306    My first graders are eager to learn about the ...
Name: essay, Length: 109248, dtype: object
```

Adding a new feature Number of words in essay

```
train["essay_word_count"] = train['essay'].str.split().str.len()
```

```
#another way to get count
c=[]
for i in train['essay']:
    a=len(i.split(' '))
    c.append(a)
```

```
train.head(2)
```

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

Train Test Split

```
y=train.project_is_approved
X=train.drop('project_is_approved',axis=1)
```

```
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.33,stratify=y, random_st
X_train, X_cv, y_train, y_cv =train_test_split(X_train,y_train,test_size=0.33,stratify=y_t
```

```
X_train.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_st
29686	160998	p011896	7e72c39ef290b9997bd13a6942da6321	Mrs	
99646	140763	p097383	ec113e80d28e9f953f2858b82fed3ef8	Ms	

Text Preprocessing

```
# printing some random reviews

print(train['essay'].values[0])
print("="*50)
print(train['essay'].values[500])
print("="*50)
```

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classroom as well
==================================================
As an RSP (resource specialist), my \"typical day\" varies depending on the day of th
==================================================
```

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

def decontracted(phrase):
    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


# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're",
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'hi
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', '
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', '
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over'
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', '
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'd
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn'
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn
            'won', "won't", 'wouldn', "wouldn't"]
```

```
sent=decontracted("\r\n\r\nEvery week, new technologies emerge that can't could engage stu
sent=sent.replace('\r',' ')
sent=sent.replace('\n',' ')
sent=sent.replace('\"',' ')
sent
```

```
'    Every week, new technologies emerge that can not could engage students and trans
```

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

```
 Every week new technologies emerge that can not could engage students and transform
```

```
preprocessed_essays_train = []
for sentence in tqdm(X_train['essay'].values):
    sentence=' '.join([i for i in sentence.split(' ') if i.lower() not in stopwords])
    sent=decontracted(sentence)
    sent=sentence.replace('\r',' ')
    sent=sentence.replace('\n',' ')
    sent=sentence.replace('\"' ,' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sentence)
    preprocessed_essays_train.append(sent.strip())
```

```
100%|██████████| 49041/49041 [00:25<00:00, 1887.37it/s]
```

```
preprocessed_essays_train[3]
```

```
'students incredibly passionate group 11 14 year olds work hard every single day desp
```

```
preprocessed_essays_cv= []
for sentence in tqdm(X_cv['essay'].values):
    sentence=' '.join([i for i in sentence.split(' ') if i.lower() not in stopwords])
    sent=decontracted(sentence)
    sent=sentence.replace('\\r',' ')
    sent=sentence.replace('\\n',' ')
    sent=sentence.replace('\\"' ,' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sentence)
    preprocessed_essays_cv.append(sent.strip())
```

```
100%|██████████| 24155/24155 [00:12<00:00, 1865.85it/s]
```

```
preprocessed_essays_test = []
for sentence in tqdm(X_test['essay'].values):
    sentence=' '.join([i for i in sentence.split(' ') if i.lower() not in stopwords])
    sent=decontracted(sentence)
    sent=sentence.replace('\\r',' ')
    sent=sentence.replace('\\n',' ')
    sent=sentence.replace('\\"' ,' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sentence)
    preprocessed_essays_test.append(sent.strip())
```

```
100%|██████████| 36052/36052 [00:19<00:00, 1874.30it/s]
```

Preprocessing of project title

```
# printing some randomproject titles.
print(X_train['project_title'].values[0])
print("="*50)
print(train['project_title'].values[150])
print("="*50)
print(train['project_title'].values[1000])
print("="*50)
print(train['project_title'].values[20000])
print("="*50)
```

```
Using Mini Technology to Get Maximum  Results!
==================================================
Building Blocks for Learning
==================================================
Empowering Students Through Art:Learning About Then and Now
==================================================
Health Nutritional Cooking in Kindergarten
==================================================
```

```
title = decontracted(X_train['project_title'].values[46])
```

```
title
```

```
    'I can make choices!!'
```

```
X_train['project_title'].values[45:50]
```

```
array(["Let's Get Moving!", 'I can make choices!!',
       'Help Us Build Our Classroom Library!',
       'Global Collaboration with Books',
       'STEM for Second Grade Scientists!'], dtype=object)
```

```
preprocessed_titles_train = []
for sentence in tqdm(X_train['project_title'].values):
    sentence=' '.join([i for i in sentence.split(' ') if i.lower() not in stopwords])
    sent=decontracted(sentence)
    sent=sentence.replace('\r',' ')
    sent=sentence.replace('\n',' ')
    sent=sentence.replace('\"' ,' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sentence)
    preprocessed_titles_train.append(sent.strip())
```

```
100%|██████████| 49041/49041 [00:01<00:00, 43359.29it/s]
```

```
preprocessed_titles_cv= []
for sentence in tqdm(X_cv['project_title'].values):
    sentence=' '.join([i for i in sentence.split(' ') if i.lower() not in stopwords])
    sent=decontracted(sentence)
    sent=sentence.replace('\\r',' ')
    sent=sentence.replace('\\n',' ')
    sent=sentence.replace('\\"' ,' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sentence)
    preprocessed_titles_cv.append(sent.strip())
```

```
100%|██████████| 24155/24155 [00:00<00:00, 42926.17it/s]
```

```
preprocessed_titles_test = []
for sentence in tqdm(X_test['project_title'].values):
    sentence=' '.join([i for i in sentence.split(' ') if i.lower() not in stopwords])
    sent=decontracted(sentence)
    sent=sentence.replace('\\r',' ')
    sent=sentence.replace('\\n',' ')
    sent=sentence.replace('\\"' ,' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sentence)
    preprocessed_titles_test.append(sent.strip())
```

```
100%|██████████| 36052/36052 [00:00<00:00, 43891.67it/s]
```

```
preprocessed_titles_test
```

```
['Math Center iPad',
 'SUPER Scientist Week r n',
 'Boombox Special Needs Music Students',
 'Sparking Discussion Current Events',
 'Toot Horn',
 'Ecosystem Challenge',
 'Go Robotics Programming FLL',
 'Targeting Kid Inspired Seating',
 'Bouncy Bands Active Learners',
 'Literature Building Engineering',
 'iPad Pro Newly Created Maker Space',
 'Code Us SIMS',
 'Empathetic Learners',
 'Safety Supplies needed STEM classroom',
 'Quiet Clean Steals Scene',
 'Put Drama Dramatic Play',
 'Wanted iPad',
 'NEW technology NEWton Falls',
 'IMPROVING LANGUAGE LITERACY',
 'Engineering Jewels Wild',
 'Digital Classroom Digital Learners',
 'Freeing Special Needs Students Clunky Old Desktop Computers',
 'Readers Paradise',
 'Classroom Chromebook Initiative',
 'Inquiring Hands Minds r nFifth Grade Alive',
 'Grab Bag Provide Great Year',
 'Watch flip flip watch read magazine',
 'Sparking Creative Writing Imaginative Word Work',
 'Flexible Seating',
 'BOOKS BOOKS READ',
 'Flexible Seating Mrs Welch s Class',
 'Learning Active Classroom',
 'r nEmbracing Diversity Using Technology Enhance Communication r n',
 'Reading Common Core Good',
 'Supplies Learning',
 'Tech Upgrade',
 'See Science Visual Aids Enhance Hands on Experience',
 'School Supplies Help Motivate Students',
 'Buzz Correct Answer',
 'Table This',
 '7th Grade Class In Class Kitchen Farm Table',
 'Creating Readers Future',
 'Squish it Squash It Mold It Stamp It',
 'Let s Code',
 'Read Books Me',
 'Mrs Miller s Second Grade SUPER READERS',
 'Learning Fun',
 'Summer Learning Tote ally Possible',
 'You Want To Not Bring Books Class',
 'Fabulous 4th Graders Request Supplies',
 'Hear It Read It',
 'Current Events Magazines Contribute Creation Competent Citizens',
 'Discovering Personal Legend The Alchemist',
 'Wiggle Seating Active Students',
 'Magnetic Blocks Exciting Learning',
 'Entrepreneurial Approach Financial Literacy',
 'Digital Learning Chromebooks',
 'Robotics Future',
 'Yoga Success',
 'Photography Class American Stories Immigration',
 'Flexible Seating Learning',
 'Write On',
 'Building Team Work Sports',
 'CONTAINers CONTAINing Centers',
 'Teaching Time Kids',
 'Cold Science',
 'OMG Books',
 'TI 84 Graphing Calculator Math',
 'Classroom Supplies Pre K',
 'Book Bin Bonanza',
 'Showing PRIDE Martin Market',
 'Books Engage Individual Student Interests',
 'Going Chrome Search Exciting New Adventures',
 'Flood Class Technology',
 'Puzzles Please',
 'Backpacks Brighter Future',
 '10 Prize Winning TOON Graphics Middle Grades',
 'Wiggle Free Learning Wobble Stools',
 'iCan iPad',
 'Learn Draw Create',
 'Sitting Stack Stools Cool',
 'Putting Technology Hands',
 'iLearn iPads',
 'Learning Exploring',
 'Crazy Science',
 'iLearning iPads',
 'Excellent Microphone Exceptional Students',
 'Dream It Draw It',
 'Help Encourage Creativity Pre K',
 'Exploring Wonder r nof reading together',
 'Slouchy Chairs Bad Orchestra III',
 'iLearn iPads',
 'Garden Grows II',
 'iPad Classroom',
 'Fairy Tales STEAM r n Science Technology Engineering Art Math',
 'Desperate Basic Supplies',
 'Math Fact Madness',
 'Shredding Oldies',
 'Building comfortable classroom',
 'Learning Groups Classroom',
 'Reader s Workshop Support Needed',
 'Chromebooks Ms Case s Classroom',
 'Wobble Wiggle Work',
 'ESL needs resources enhancing learning',
 'Technology Needs Special Needs Students',
 'Readers TOON Graphic Novels',
 'Technology Help Learning r nScience Math',
 'Words Come Alive Audio Books',
 'Wiggle Work',
 'Fit Lab',
 'Important Behavioral Classroom Incentives Supplies',
 'Flexible Seating Blended Learning',
 'Normal Overrated Analyzing Data Finding Trends Making Projections',
 'STEAM Bilingual Education',
 'Wyoming History Adventures 4th Grade',
 'Backpacks Connect Home School',
 'Flexible Work Areas Busy Second Graders',
 'Exercise Brain',
 'Ready Reading',
 '2016 2017 School Supplies',
 'LEGO simple machines Kindergarten',
 '21st Century Learning Career College Readiness',
 'Spring 2017 Female Sports Health Workouts',
  'Spring 2017 Female Sports Health Workouts',
 'Listening Ears Learning Minds',
 'Get Out Get Active PLAY',
 'Mind Benders Innovating Educating One Game Time',
 'Total Body Fitness Sports Challenge',
 'Connecting Chromebooks',
 'Chromebook Management Fun Math',
 'Charge Up',
 'Mystery Disease',
 '21st Century Literacy Success',
 'Information Drives Success',
 'Robotics Challenge',
 'Chromed',
 'Technology Classroom',
 'Strings All Part 1',
 'Clicking Away',
 'Almost Famous Musicians Making',
 'SMART Interacting Music',
 'Alphabet Resources 4K',
 'Apple Today Tech2Teach Needs',
 'Charting Ideas',
 'Flexible Seat Can t Beat',
 'Visual Learners Need See Future',
 'Economics TIME it Economics Society',
 'Classroom Dictionaries',
 'Sitting Option Students choosing tools help learn better',
 'Hear Now',
 'Read it Write it it',
 'visual organization children work independently',
 'Let s Bring Technology 4th Grade',
 'Drink Pink Ink',
 'Learning Read Write Goal',
 'Connect Roam Chrome',
 'Window World',
 'Let s Find Out World',
 'Improving Focus Classroom Movement',
 'Time Read Non Fiction',
 'Keep Moving Keep Learning Keep Reaching Stars',
 'Reading Liftoff',
 'Flexible Seating Come',
 'Learning Matters Social Emotional Resources',
 'Juiced Learning',
 'Dynamic Cart',
 'Flexible Seating Creates Success',
 'Flexible Seating Back Again',
 'Warm Rested Ready Learn',
 'Lego Education WeDo Software',
 'Wobble Baby Wobble Baby Wobble Baby',
 'Hit FLEX',
 'Eat Breathe Cheer Repeat',
 'STEMing learning brains fairy tales',
 'TOON Tastic',
 'Global Interactive Learning Community',
 'Help us get hand wipes',
 'Classroom Learning Materials',
 'Wild Centers',
 'Making Reading Fun',
 'Dig It',
 'Chrome Book Worms',
 'Healthy Snacks will boost Brainpower',
 'What s Going World',
 'Learning Economics Technology Kindergarten',
 '5th Grade Engineers say',
 'Please provide URINE specimen',
 'Writing Applying Research',
 'Explore express learn',
 'Games Increase Critical Thinking',
 'Science Tools Hands On Learning',
 'Sit In Teaching Drive In Brains',
 'iPads Enhance Student Learning',
 'Genius Hour Project Kids',
 '360 Camera',
 'Make ing Space Organize Materials',
 'Charged READY learn',
 'Core Strength Classroom',
 'world water',
 'Art STEAM Fuel',
 'Manga makes world go round library',
 'Green Books Green Kids',
 'Stool Perch Upon',
 'Organized Beginning',
 'Hear Now',
 'STEM Learning Young Einsteins',
 'Simple Things Life',
 'Books Love',
 'Middle School Band Needs Speaker',
 'Bounce Success',
 'Bringing fresh vegetables STEM classroom',
 'Smelly Spelly',
 'Magic Carpet Ride Learning',
 'Portfolios Second Graders',
 'Volley Health Wellness',
 'Wobble Chairs Active Students',
 'Remediate Chromebooks',
 'Class Book Set Tales Odyssey Part 2',
 'Miss Luce s Classroom Mailbox',
 'Alleviating I Dont a Desk Pouches',
 'Imagination Comes Life',
 'Movement Music Fitness',
 'Mound Books',
 'Music Supplies Another Successful Year',
 'Chromebooks Music Composition',
 'Tickling Ivories',
 'Want Kids Without Engaging Books',
 'Bouncing Kinder',
 'Classic Economics Texts AP Classroom',
 'Durable Folders High Needs Learners',
 'Building Skills Laptops Learning',
 'Caution Learning Progress',
 'Exploration Senses',
 'Using Stem Projects Help Young Minds Flourish',
 'Balsa Wood Gliders MESA Project',
 'Interactive Learning Materials Engage Students',
 'Teaching Student Autism Work Independently',
 'Let s Fun Activity Table',
 'Shelf House BOOK',
 'TK Build It',
 'Living Year Books',
 'Close Reading Kits',
 'Photography Classroom Needs Lens Digital Cameras',
 'Fifth Grade Whizzes Need Computer',
 'Building Readers One Game Time',
 'Learning Time Family Time',
 'Hitting Targets Helping Students Get Grade Level',
 'Hitting Targets Helping Students Get Grade Level',
 'Blending Math Technology Flipped Classroom ATPE',
 'Hey Kids Let s Go Green',
 '21st Century Mathematicians',
 'Shine Loud 4th Grade Math Students',
 'Learning Art',
 'Time Play',
 'Active Minds Bodies Flexible Seating Successful Students',
 'Science Center Supplies',
 'Mice Running Empty Part II',
 'Wiggle Work',
 '21st Century Learning Multimedia',
 'Help Ear Buds Falling Out',
 'Fall Tech Knowledge y',
 'Ink Dink YOU',
 'Write Stuff',
 'Magazine Build Interest Skill',
 'Success Reading',
 'Sit Collaborate Listen Aim Make 8 Additions',
 'Bouncing Learners',
 'Character building books 1st 2nd graders',
 'New Computer Creative Media Students',
 'Spiking Way Comfort Zones',
 'Becoming Better Readers Writers',
 'Growing Readers One Book Time',
 'Badminton Time',
 'Picking Right Seat ME',
 'Math Math Math',
 'Take Look Donate Book',
 'Looking Monsters Create Fun',
 'TIME FUN GAMES',
 'One One Technology Best Student Achievement',
 'Engaging Arts Books',
 'Dramatic Play Activating minds educational excellence',
 'Strumming Beat',
 'Planting Seeds Knowledge Third Grade',
 'Little Readers',
 'Telling Stories Writing',
 'Flexible Seating Flexible First Grade Learners',
 'Technology Growing Minds',
 'Basic Ceramics',
 'Reading Success',
 'Read Harry Potter',
 'Alternative Seat Choices',
 'Backpacks Leaders',
 'Keeping Arts Alive',
 'Move Learn',
 'Focus Support',
 'Creating Classroom STEM STEM',
 'STEAM Students Need Daily Technology',
 'See World Grain Sand',
 'Ice Black Eyes',
 'Montessori Materials Support Mixed Age Classroom',
 'Chromebooks Communication',
 'Building Engineers Kindergarten',
 'Reading Comprehension',
 'Help us get Healthy',
 'Tech Fabulous First Graders',
 'No Stress 4th Grade Desks',
 'Bossy R Not Going Boss Us Around',
 'Interest us Lose us',
 'STAAR Test Ready',
 'Bringing Color Projects',
 'southern teacher needs horseshoe table y all',
 'Wiggle Move Learn',
 'Kinder Counts',
 'Need Bounce Wiggle Stay Target',
 'Fun Outdoors',
 'Growing Mathematicians',
 'Best Bee r nUtilizing Technology ST Math',
 'Save Tree Spare Marker',
 'Bountiful Binders',
 'Let s bounce learning',
 'Students Soar Parents Support Sumdog Summer Success',
 'Destination America r nLights Camera Action',
 'Creating Air',
 'Backpacks Little Learners',
 'Building Brains ONE Bridge Time',
 'Yoga Lifelong Healthy Life Style',
 'Bigger Screens Better Work Efficiency',
 'AP World History Exam Review Books',
 'Big Time Reading Kindergarten',
 'Media Literacy Teaching Current Events Upfront Magazing',
 'Creating Positive School Culture Incentives Rewards',
 'Battle Books Sunshine State Book Club',
 'Tech Tech Everywhere But Put it',
 'Chromebook High Poverty High School English Students',
 'Learning Stop Motion',
 'Need Puters',
 'Pride Field',
 'Assistive Technology ESE Autistic Classroom',
 'Art Achievement Tigers',
 'Hands On Science Tiny Hands',
 'We re putting A STEAM Full STEAM ahead',
 'Visualize learning Helping students see science',
 'Lead Free Water Kindergarten Students',
 'Music Books Play',
 'Cow Eyes',
 'Technology Classroom Teaching Students New Age',
 'Counting Steps Future Healthy Fit Lives',
 'Want Scientists STEM Classroom',
 'Keeping Kids Learning Good Organization',
 'Targeting Healthy Snack Consumption',
 'Technology Future',
 'Back Future',
 '21st Century Kids Need 21st Century Tools',
 'Making Difference Small Group Instruction',
 'Basketballs Footballs Oh My',
 'Enthusiastic Teacher Seeks Supplies Deserving Students',
 'Fun Enaging Learning Centers',
 'Sensory Tools Special Education Students',
 'There s No Place Like Child s Pose',
 'Fitness Trackers',
 'Making Special Education Special',
 'iLab SLE',
 'Strumming Success',
 'Taking Hot Air Education',
 'Writing Style',
 'Classroom Library Fun',
 'Sensory Smart Classroom',
 'Love Learning Centers',
 'Reading Math Manipulatives Rock',
 'Learn iPad',
 'Bring Coding Home',
 'Counting Money Resources',
 'Technology Everyone',
 'need Wiggle Wobble Work',
 'Making Math Happen',
 'Learning Always Sit Desk',
 'Technology Key Success',
 'Take Us Organization Station',
 'Cray Pens Creating',
 'Listening Center',
 'Wild World Around Us',
 'Little Scientist Learn Discovery Exploration',
 'Magical Adventure Awaits',
 'Kindergarten Market',
 'HELP NEED COMPUTERS LOVE SCHOOL LEARNING',
 'Books Needed Eager Readers',
 'Center Supplies Transitional Kindergarten',
 'Love Learning Read',
 'Title Students Need Advanced Calculators',
 'Puzzle Mat',
 'Establishing Positive Behavior Financial Literacy Classroom Store',
 'Wish Upon Ukulele Star',
 'Scientific Cut Rest',
 '5th Grade Goes Digital',
 'Living Habitats Breathing Animals Engaged Learning',
 'Calculators Dobson 3 5',
 'Hear Now',
 'Wiggle Jiggle Work',
 'can t see',
 'Wiggle Wobble Learn',
 'Headphones Essential Maximize Learning',
 'need printmaking supplies',
 '1st Step Towards Google Classroom',
 'Improve learning comfy environment',
 'Listening Station Improve Fluency',
 'Conservation Corps Portraiture',
 'Language Centers Needed',
 'Downright Smashing',
 'Bean Bags Thoughts',
 'Watch wobble sit',
 'Magic Carpet Ride',
 'Extra Extra r nRead It',
 'Let s Give Em Something TALK About',
 'Fantastic Novel Study',
 'Motor On',
 'Technology Spark Criticial Thinking',
 'Back School Basics',
 'Less Distractions Learning',
 'Reeds cost much',
 'Lets Learn English',
 'Little Movement Strive Potential',
 'Pottery Desert ATPE',
 'Colored Chairs READ 180 Groups',
 'Tech Tykes',
 'Hands on Fun',
 'Rockin Rollin Learning',
 'Batteries New Kid Safe Appliances',
 'My Students Want FUN',
 'Bright Color Cats',
 'Tech Savvy Kindergarteners',
 'Economic Decisions Google',
 'Instruments Need Home',
 'Life Essentials Students',
 'iPad 1st Grade',
 'Three cheers Flex Seating',
 'Ready Set Read',
 'Paint Brushes Paint Talented Young Artists',
 'Balance Discs Get Students Moving Learning',
 'Positive Productive Learning Setting Help Struggling Students',
 'Folders Conferences',
 'Charged Up',
 'Grade Level Guided Reading Literacy Books',
 'Differentiated Math Folders High Needs Learners',
 'little Sensory Assistance go Long Way',
 'Math Games Everyone',
 'Lights Learning',
 'Tech Titans',
 'Plan it Build it',
 'Laser Focus Sticky STEM',
 'Need Ink Help Us Think r n',
 'That s SHOCKING 5th Grade Science',
 'Comic Books',
 'Suceeding MathTECHmatics',
 'Technical Math Manipulatives',
 'Readers Need Engaging Empowering Books',
 'Flexibility Focus Fourth',
 'Imagination 3D Characters',
 'Passionate Percussion',
 'Nutrition What s Left Bayou Part 2',
 'Wonderful World LEGO Ozobot',
 'Computers needed please',
 'hands Deck',
 'Life Essentials Builds Self Esteem',
 'One book box time',
 'Bringing Real World Classroom',
 'STEM Family Literacy',
 'Ready Ball Kindergarten',
 'Learning Healthier Classroom Community',
 'Exercise Strong Bodies Brains',
 'Hear Now',
 'Lamination Station',
 'iPad Bliss',
 'STEM Art We re starting STEAM',
 'Ready Best Be',
 'Hocus Pocus Everyone Focus Flexible Seating Firsties',
 'German Student GoPro Video Soccer League Film Making Project',
 'World Around Us Kids',
 'Technology Engagement',
 'Spectacular Centers',
 'Sensory Seating Opportunities',
 'Helping Community one step time',
 'need help becoming digital learners',
 'Chefs Future',
 'Flexible Seating Options',
 'Technology Tip Fingers',
 'Fitness Achievement Recess',
 'Kindle Fire Learning Tablets',
 'Little Innovators Need STEM Tools',
 'Wonderful World',
 'Need Listening Center',
 'Wiggling Way Higher Achievement',
 'Keeping Calm Work',
 'Super Seats Sounds Second Graders',
 'Mini iPads Big Learning',
 'Creativity Expanded Using 3Doodler Pens',
 'Planners Success',
 'iPad Introduction',
 'Culture Autism',
 'Staying Organized',
 'VR Classroom Make real kids',
 'Desperately Seeking Dollars Technology',
 '3Doodler 4Creating',
 'Technology Kindergarten',
 'Coat hooks cubbies',
 'Rebuild Sight',
 'Giving Hearts',
 'Books Beyond',
 'F Fruit Go',
 'Music Stands Musicians',
 'Stepping Learning Kindergarten',
 'Successful Math Centers',
 'Basic Needs Please',
 'Beyond Textbook 2',
 'Extra Extra Read It',
 'Chrome Grow',
 'Let Move It Move It',
 'Drums Stands Djembes',
 'Using Technology Gain Knowledge',
 'Spot Everyone',
 'First Grade SUPER STAR Readers',
 'STEMriffic learning',
 'Wired Work',
 'Let Autism Speak',
 'Artists',
 'Operation Chromebooks 2 0',
 'Point Math Reading',
 'Exploring Science Makey Makey Kits',
 'Task Math Class',
 'Beating Balancing Odds Comfy Cushions Determination',
 'Getting Students Fired Spanish',
 'Osmo Opportunities',
 'Books ALL Student Readers Teacher Read Alouds Needed',
 'Read IT',
 'Kindergarten Fun',
 'Sit Here',
 'Thanks Kindness',
 'Instruments Need Cleaned',
 'Seeing Eyes POET',
 'Keying Reading',
 'Help Us Make Art Social Studies',
 'Physics Interactive Notebook Supplies',
 'Math not worksheet',
 'Lets Make Masterpiece',
 'File Folder Fun Math Science Social Studies',
 'Stability Balls Keep Us Rolling',
 'Oh No Allergy Season',
 'Active Learners',
 'Ipad Minis Kindergarten',
 'STEM starts home',
 'Sprucing Classroom Library',
 'Read Write Tiles',
 'Type Write Holiday Cheer',
 'Zeal math tutoring',
 'Kinder s Kangroos get creative technology',
 'Hear Now',
 'Students Take Ownership Learning Technology',
 'Making Math Science Interactive Technology',
 'Changing World One Picture Time',
 'League Own',
 'Hear Now',
 'RMS STEM',
 'Learning Code Future',
 'One to One Adventure APUSH AP Psychology',
 'Helping Us Grow',
 'Fidgit Focus',
 'Learning Already Love Reading',
 'CLEAN COLORFUL CLASSROOM CARPET',
 'Space Final Frontier r nMore Table Space Learning',
 'Second Graders Reaching Goals',
 'Bring Tech Availability Challis',
 'Technology Matters',
 'Focusing Hokki Movement',
 'Roaming Reporters',
 'Focused Success',
 'Power Recess',
 'Excursions Drawing',
 'Everything Better Chrome',
 'Eager Students Want Continue Learning',
 'Building Blocks Pre K',
 'Alternative Classroom Seating',
 'Expanding Horizons Virtual Field Trips',
 'Indoor Recess Blues',
 'Art supplies needed Kindergarten Learning',
 'Let s Read It',
 'Supplies Responsibility r nConfident Learners',
 'Flexible Seating',
 'Wiggle Work Learn',
 'Ready Set Record',
 'Stand Success',
 'Full STEM Ahead',
 'Writing Workshop Open',
 'picture worth thousand words',
 'Advanced Placement students seek Song Solomon',
 'Learning Photography Early Age',
 'Sparking Fire Kindergarten',
 'Integrating Technology',
 'Tech Advantage',
 'Headphones Needed',
 'Math stations',
 'Sharper Minds',
 'Writers',
 'Mathematics Finding X Not r nOnly Pirates',
 'Sitting Day Much',
 'Fine Tuning Daily 5 Right Stuff',
 'Choice Destination',
 'Let s Explore World Around Us',
 'Scientific Researchers',
 'Maker Movement',
 'Achoo Let Help You',
 'Organized Class Successful Kids',
 'Supplies Learning',
 'Mindful Middleschoolers',
 'Chromebooks 6th Grade',
 '21st Century Learners Need Backpacks',
 'Creating Critical Thinking Chromebooks',
 'Reading Meaning',
 'Hands Math Home',
 'Electric learning Makerspace',
 'What s Dinner',
 'Hands Optics Project',
 'Activity Bots Student Programming Education',
 'Coding Carnegie',
 'Chemistry Conflict',
 'Rex Bunny Needs Love',
 'Accessing Common Core',
 'Listen Up',
 'Get Blood Pumping',
 'Search Search Search',
 'Better Working Conditions Loom Ahead',
 '29 exciting books read',
 'Sport Balls Recess Keep Students Moving',
 'Pushing Pulleying Physics',
 'Tablets Reading Interventions Please',
 'Sometimes Better Equipping Young Learners Engaging Technology',
 'Building Readers Technology',
 'Environmental Engineering 3D Printing STEM Solutions',
 'Making Ideas Stick',
 'Yoga Paves Success Whole Child',
 'H Okay Need Hokki Stools',
 'Keeping Organized',
 'Balancing Scales',
 '20 Copies Lightning Thief Graphic Novel',
 'Science Reading',
 'Full STEAM Ahead',
 'Health Hard Live Without',
 'Flexible Seating Choices',
 'Love Good Series',
 'Expressively Creative',
 'Flexible seating young active learners',
 'STEM tastic Students Need Notebooks',
 'Literacy Chemistry',
 'Podium Notebooks Help Students Shine',
 'Easel Can Read Write Learn Together',
 'Let s Connect',
 'Partnering Parents Leads Student Success',
 'Nice Place Sit',
 'Chromebooks STEM students',
 'Chromebooks Leaders Tomorrow Cont',
 'Keeping us Charged',
 'Light Up STEAM Learning E textiles',
 'Using Touch Technology Advance Preschool Education',
 'RealWorldMath Teens',
 'Please Help Us Find Missing Pieces',
 'LAPTOPS LEARNING',
 'SIT LEARN',
 'Ink Classroom Printer Pretty Please',
 'YES YEARBOOK',
 'Making Writing Reading Fun 4th Graders',
 'Next Generation Writing Tools',
 'Fun Learning English',
 'Create Play Learn OSMO iPads',
 'Techno Tablet Kinders',
 'Motivate Educate',
 'Cozy Band Room',
 'Building Kinder Core Strength',
 'Wiggle Work',
 'Fill house stacks books Dr Seuss',
 'Art Build Bridges Across World',
 'Little Bits Inspire Big Ideas',
 'Little Bits Inspire Big Ideas',
 'Redesigning Classroom Environment',
 'Bass',
 'Three Cheers Homework',
 'Need Art Supplies',
 'iPads Foster Communication',
 'Books Books everyone',
 'Mapping Reading Space',
 'Healthy Teeth Happy Student',
 'Opening world endless potential Reading',
 'Listening Learn',
 'Comfy Seating Home Away From Home',
 'One Book One Child',
 'Take Look Now',
 'Living Organisms Exploration',
 'Need Reboot',
 'Reading Science',
 'Organize Chaos',
 'Tech Time',
 'Elf Shelf',
 'Robotic Art',
 'Music Math Muscle',
 'Help Resurrect Shakespeare Romeo Juliet Books Freshmen',
 'Kindergarten Needs Sensory Play',
 'Laptops Little Learners',
 'Tracking way Fitness',
 'Wiggling Success',
 'Search World Answers',
 'Let s Get Healthy',
 'Playtime important classroom time',
 'Learning Read',
 'Technology Takes Us Places Spaces Effectively Use Classroom Technology',
 'TOON books Inspiring LOVE Reading',
 'Thirsty Technology',
 'Supplying Success',
 'Give Player Soccer Ball',
 'Charging Education',
 'First Grade Reading Stars',
 'Right Target',
 'Literature Study HMS',
 'EBD Special Education Sensory Items',
 'Maker Space Learning Place',
 'Flexible Seating Deserving Students',
 'Let s Get Organized Comfortable',
 'Ready Set READ',
 'Get Wiggles Out Seating Options Include Motion',
 'Hands on Math Centers',
 'iPad Mini Two Go Long Way',
 'Blooming Apple',
 'Rockstar Reading Resources',
 'Kid Inspired Book Choices',
 'EV3 Lego Robotics Real World Programming Engineering',
 'Connecting Learning PreK Interests',
 'Students Crazy Apple Watches r n',
 'Classroom Supplies Books Galore',
 'Chromebooks 2nd Grade Stars',
 'Creative technology',
 'Bump Up Set Up',
 'Practice Make Perfect',
 'Cultivating Book Worms',
 'Listen Learn',
 'Different Kind Treat Job Well Done',
 'Exceptional Students Need Supplies Music Performance',
 'Brush Way Wonderful',
 '5th Grade Googlers Seeking Chromebooks Part 2',
 'Osmo Means Interactive Learning FUN',
 'Grab Book and Let s Get Comfy',
 'Getting Techie It',
 'Chromebooks reading writing',
 'Matter Privacy',
 'Make Lesson Accessible North Minneapolis Young People',
 'Keep Moving Musical Instruments',
 'Art Language Printmaking Voice',
 'Healthy Lifestyles',
 'Chart Flippin Class',
 'Help Us Succeed Reading',
 'Refueling Learning',
 'Full STEM ahead',
 'Osmos Opening Imaginations Minds',
 'Super Scientist',
 'Making Learning Come Life',
 'Meeting Area',
 'Playground Equipment Help Students Keep Fit',
 '21st Century Technology 2nd Graders',
 'Pre K Centers Fun Engaging Purposeful',
 'Printer copier scanner music room',
 'Learn Pads',
 'Want Remember',
 'Back Basics',
 'Literacy Listening',
 'Technology Efficiency',
 'Virtual Exhibition Virtual Reality Student Exhibition Lives',
 'Building Class Set Computers',
 'Crayons Pencils Supplies Oh My',
 'Making Maker Space',
 '3rd Grade Special Needs Class',
 'Tablets Kindergarten Students',
 'Shoulder Rests Outstanding Orchestra',
 'Creative CriCut Art Integration',
 'Help Keep Us Track',
 'OSMOS Making learning hands on',
 'Fill Em Good Fuel',
 'Write Stuff',
 'Music Ears',
 'Empowering Pre K Students Art',
 'Learning Play Hockey',
 'Comfy Collaboration',
 'Flexing Learning Muscles Flexible Seating',
 'Big Picture',
 'Little Readers Sitting Together Library Area',
 'Wanted Pencils Basic Supplies Oh',
 'Frisbee Golf Gulp',
 'One School One Book',
 'SCOPE ing Fluent Readers',
 'Preparing Success',
 'Chromebooks Essential Tool Developing Success',
 'Storage Solutions',
 'Pre K Rocks Part 2',
 'Chromebooks Removing Barriers Technology',
 'Ideal Learning',
 'Help Us Find Stuff',
 'Great Migration Comes Alive',
 'classroom need headphones please',
 'Sphero s Phys Education class',
  'Sphero s Phys Education Class',
 'Place Gather',
 'Kill Home Energy Waste',
 'Essential Healthy Snacks Starving Low Income Students',
 'Rollin Gold Medal Part 2',
 'Hands on Clean',
 'Moving Recess',
 'Dissecting Physiology',
 'Ink Me Printer Ink Needed',
 'Target Fitness Goals Part II',
 'Computers Change Classroom',
 'Today Exploring Science Classroom Tomorrow Saving World',
 'Supplies Demand',
 'Pangaea Plate Tectonics Hands On Learning',
 'Learning Stools',
 'Comfotable Inviting Classroom',
 'Colorful World',
 'Video Mics Student Written TV Series',
 'Help Us Protect Hear Kindles',
 'Book Bins CRT Materials Needed',
 'Get Fire d Reading',
 'Keep Change Kindergarten Guide Financial Literacy Part 2',
 'Let s Get Boys Reading Amazing Books',
 'Nooks Books',
 'Mission Move',
 'Differentiation Time',
 'Technology Help Enrich Instruction',
 'Life Lens',
 'TOON Classics',
 'Box Vocabulary',
 'Google n classroom',
 'NEATO Non Fiction',
 'Stuff Needed',
 'Creating Student Driven Learning Environment',
 'Getting Wiggles OUT r n',
 'Attention Grabbers',
 'Building Creative Minds',
 'Snacks Growing Bodies',
 'Let s Table Ideas Horseshoe Table',
 'Preparing Real Life Jobs',
 'Hip Hop Hooray',
 'Outstanding Organization',
 'First Chapter Second Grade Reading',
 'Mastering Math',
 'ABC Ya 4K',
 'Future Geographers Unite',
 'Summer Book Swap',
 'Colorful World',
 'Anything Grow Up',
 'See It Understand',
 'Hamming Hamlet Act II',
 'Love Books Finding ONE',
 'Basic School Supplies High Needs Kids',
 'I m focused change WORLD',
 'Making Mark',
 'Let s Get Fired Up',
 'Ready Read Second Grade',
 'Vamos Leer in Spanish',
 'Show It Share Mini',
 'Chubby Cubbies',
 'Working Wiggles',
 'Social Studies Simulation Fun',
 'Let s Play Teaching Social Skills Preschoolers Autism',
 'Empowering Students Art Chromebooks Room 123',
 'Body Motion Brain Motion',
 'Listening Center Needs Makeover',
 'Leveled Literacy Language Learners',
 'Investing Stock Market',
 'Wonderful Workspace Wigglers',
 'Books Please',
 'Student Designed Built Monitored r nIn Class Aquaponic Station',
 'Spelling Counting Technology Financial Literacy',
 'Annie Oakley American Legend',
 'Active Learning Hot Dots',
 'Bounce Focus',
 'Full STEAM Ahead Learning',
 'Shoot',
 'Keep calm help us get supplies Pt 1',
 'Hit Target alternative seating',
 'Building Better Learning',
 'Help Special Education Class Movement',
 'Winter Holiday Reading Home',
 'Expanding Biblioteca Part 2',
 'Help Us Hands On',
 'Step Step 4 Healthy Lifestyle',
 'Interpreting Characters Tiger Rising',
 'Students Need Basic Essentials',
 'Becoming Media Literate',
 'Alphabet games',
 '3 D Print It',
 'Finding Path Tech',
 'CHARACTER COUNTS',
 'Good Mornings Great Days',
 'Cozy Carpet Cool Kinders',
 'Stepping Kindergarten',
 'P E Tech',
 'Start Spreading News',
 'Wiggle Wobble Way Flexible Learning',
 'Full STEAM Ahead Tools Designers Tomorrow',
 'Extending Learning Beyond Classroom',
 'Woodworking Sanders',
 'Water Bottle Filling Station Helps Thinking',
 'Ready Set Read',
 'Keeping Current Events',
 'Hands Learning',
 'Visual Icons Visual Learners',
 'Run Jump Play Learn',
 'Language Literacy Carpet',
 'Classroom Carpet Supplies',
 'Plugged Good Book',
 'Integrating Literacy STEM',
 'Learning Movement Exercise',
 'Need Yoga',
 'Healthy Snacking Mindful Eating r n',
 'Can t Stop Needing',
 'Flexible Seating Focusing Minds',
 'TIME Kids Read News',
 'Critters Exploration',
 'Help Us Healthy Butterfly Habitat',
 'Building Engineers',
 'Let s Make Learning Fun',
 'Paper Laminator',
 'Literature Seniors Love',
 'Technology Tablets Children Special Needs',
 'Technology Tablets Children Special Needs',
 'Using Dell Chromebook Tech Social Studies Classroom',
 'Love Reading',
 'Brain Games Future Brainiacs',
 'Expanding Knowledge Technology',
 'Mastering Math Manipulatives',
 'Adding Learning Process',
 'Environment Math Art',
 'Bitten Bocce Ball Bug',
 'Place Sit Write',
 'Leveled Books Growing Minds',
 'Mapping EDucation',
 'Making Mark World',
 'BOOKs Building Knowledge',
 'Time Magazine Subscriptions Library',
 'Chromebook Classroom',
 'Signing Day Dominguez High School',
 'Let s learn hands on',
 'Flexible Seating Personalized Learning Environment',
 'Everyday Day STEM Activities',
 'Play Powerful Learning Tool',
 'Magnificent Music Makers 2 0',
 'Fueling smart choices',
 'Reading Writing News',
 'Full STEM Ahead',
 'Technology Third',
 'Graphic Novels Nuclear Chemistry',
 'Help Us Pack Up',
 'Help kids LOVE reading',
 'Let s Dance Wiggle Stretch',
 'Virtually Running History',
 'Light Lessons',
 'Technology Rescue',
 'Books Need Home',
 'Advancing world technology',
 'No Clutter',
 'Engaging Students',
 'ESOL Bouncing Literacy',
 'Mile Runners',
 'Help us Explore Google Chromebooks',
 'Place Everyone',
 'Hablamos espa ol Chromebooks Authentic Spanish Langauge Experiences',
 'Let s Move',
 'Virtual Science Classroom',
 'Kindergarteners Love Chicken Dance',
 'Help iPads Need Protection',
 'Ican Ipad',
 'Fun Math Activities STEAM Kindergarten',
 'Little Rockets Need get Stars',
 'Full STEM STEAM Ahead',
 'WILD weather',
 'Weighted Vest Blanket Calming',
 'Creative Arts Create Engaged Students',
 'Field Day Fun',
 'Book Bins Beginning Readers',
 'CBU tiful Learners',
 'Calming Transition Time',
 'Alphabet Carpet Squares Used Mind Body',
 'Like Move It Move IT',
 'Learning Read Practice Academic Skills Using Technology',
 'Chromebooks Literacy Learning',
 'Writing Road Reading',
 'Smartphone O Meter',
 'Child CPR Project',
 'Full Bellies Warm Hands Dry Feet',
 'Focus Pocus',
 'Shared Stories Create Connections Inspire Inquiries',
 'Book Club Readers May Tomorrow s Leaders',
```

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

X_train_state_ohe=vec_state.transform(X_train['school_state'].values)
X_cv_state_ohe=vec_state.transform(X_cv['school_state'].values)
X_test_state_ohe=vec_state.transform(X_test['school_state'].values)
```

```
print(X_train_state_ohe.shape,y_train.shape)
print(X_cv_state_ohe.shape,y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
```

```
(49041, 51) (49041,)
(24155, 51) (24155,)
(36052, 51) (36052,)
```

```
X_train['teacher_prefix'].value_counts()
vec_prefix=CountVectorizer()
vec_prefix.fit(X_train['teacher_prefix'].values)
X_train_teacher_ohe=vec_prefix.transform(X_train['teacher_prefix'].values)
X_cv_teacher_ohe=vec_prefix.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe=vec_prefix.transform(X_test['teacher_prefix'].values)

print(X_train_teacher_ohe.shape,y_train.shape)
print(X_cv_teacher_ohe.shape,y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vec_prefix.get_feature_names())
```

```
(49041, 5) (49041,)
(24155, 5) (24155,)
(36052, 5) (36052,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
```

```
X_train.project_grade_category.value_counts()
vec_grade=CountVectorizer()
vec_grade.fit(X_train['project_grade_category'].values)
X_train_grade_ohe=vec_grade.transform(X_train['project_grade_category'].values)
X_cv_grade_ohe=vec_grade.transform(X_cv['project_grade_category'].values)
X_test_grade_ohe=vec_grade.transform(X_test['project_grade_category'].values)

print(X_train_grade_ohe.shape,y_train.shape)
print(X_cv_grade_ohe.shape,y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vec_grade.get_feature_names())
print(vec_grade.get_feature_names())
```

```
(49041, 4) (49041,)
(24155, 4) (24155,)
(36052, 4) (36052,)
['3_5', '6_8', '9_12', 'prek_2']
```

```
X_train.clean_subcategories.value_counts()
vec_sub=CountVectorizer()
vec_sub.fit(X_train['clean_subcategories'].values)
X_train_clean_subcategories_ohe=vec_sub.transform(X_train['clean_subcategories'].values)
X_cv_clean_subcategories_ohe=vec_sub.transform(X_cv['clean_subcategories'].values)
X_clean_subcategories_grade_ohe=vec_sub.transform(X_test['clean_subcategories'].values)

print(X_train_clean_subcategories_ohe.shape,y_train.shape)
print(X_cv_clean_subcategories_ohe.shape,y_cv.shape)
print(X_clean_subcategories_grade_ohe.shape, y_test.shape)
print(vec_sub.get_feature_names())
```

```
(49041, 30) (49041,)
(24155, 30) (24155,)
(36052, 30) (36052,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government', 'colleg
```

```
vec_cat=CountVectorizer()
vec_cat.fit(X_train['clean_categories'].values)
X_train_clean_categories_ohe=vec_cat.transform(X_train['clean_categories'].values)
X_cv_clean_categories_ohe=vec_cat.transform(X_cv['clean_categories'].values)
X_test_clean_categories_ohe=vec_cat.transform(X_test['clean_categories'].values)

print(X_train_clean_categories_ohe.shape,y_train.shape)
print(X_cv_clean_categories_ohe.shape,y_cv.shape)
print(X_test_clean_categories_ohe.shape, y_test.shape)
print(vec_cat.get_feature_names())
```

```
(49041, 9) (49041,)
(24155, 9) (24155,)
(36052, 9) (36052,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_langu
```

Vectorizing Numerical features

Various numerical feautures are :

1.Price

2.Quantity

3.Number of Projects previously proposed by Teacher

4.Title word Count (introduced by us)

5.Essay word Count (introduced by us)

```
resourc_data=resources.groupby('id').agg({'price':sum, 'quantity':sum}).reset_index()
resourc_data.head()
```

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21
2	p000003	298.97	4
3	p000004	1113.69	98
4	p000005	485.99	8

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

```
#Price
from sklearn.preprocessing import Normalizer
X_train['price'].value_counts()

#from sklearn.preprocessing import Normalizer
nm=Normalizer()

nm.fit(X_train['price'].values.reshape(-1,1))

X_train_price_norm = nm.transform(X_train['price'].values.reshape(1,-1).T)
X_cv_price_norm = nm.transform(X_cv['price'].values.reshape(1,-1).T)
X_test_price_norm = nm.transform(X_test['price'].values.reshape(1,-1).T)


print(X_train_price_norm.shape,y_train.shape)
print(X_cv_price_norm.shape,y_cv.shape)
print(X_test_price_norm.shape,y_test.shape)
```

```
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

```
nm_posted=Normalizer()
nm_posted.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)
X_train_No_of_teachers_norm=nm_posted.transform(X_train['teacher_number_of_previously_post
X_cv_No_of_teachers_norm = nm_posted.transform(X_cv['teacher_number_of_previously_posted_p
X_test_No_of_teachers_norm = nm_posted.transform(X_test['teacher_number_of_previously_post
print("After vectorizations")
print(X_train_No_of_teachers_norm.shape, y_train.shape)
print(X_cv_No_of_teachers_norm.shape, y_cv.shape)
print(X_test_No_of_teachers_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
====================================================================================
```

```
X_train.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160998	p011896	7e72c39ef290b9997bd13a6942da6321	Mrs	RI
1	140763	p097383	ec113e80d28e9f953f2858b82fed3ef8	Ms	NJ

```
#Quantity
nm_quantity=Normalizer()
nm_quantity.fit(X_train['quantity'].values.reshape(-1,1))
```

```
X_train_quantity_norm = nm_quantity.transform(X_train['price'].values.reshape(1,-1).T)
X_cv_quantity_norm = nm_quantity.transform(X_cv['price'].values.reshape(1,-1).T)
X_test_quantity_norm = nm_quantity.transform(X_test['price'].values.reshape(1,-1).T)
```

```
print(X_train_quantity_norm.shape,y_train.shape)
print(X_cv_quantity_norm.shape,y_cv.shape)
print(X_test_quantity_norm.shape,y_test.shape)
```

```
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

```
#title_word_count

nm_title_count=Normalizer()
nm_title_count.fit(X_train.title_word_count.values.reshape(-1,1))
X_train_title_word_count_norm = nm_title_count.transform(X_train['price'].values.reshape(1
X_cv_title_word_count_norm = nm_title_count.transform(X_cv['price'].values.reshape(1,-1).T
X_test_title_word_count_norm = nm_title_count.transform(X_test['price'].values.reshape(1,-
print(X_train_title_word_count_norm.shape,y_train.shape)
print(X_cv_title_word_count_norm.shape,y_cv.shape)
print(X_test_title_word_count_norm.shape,y_test.shape)
```

```
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

```
#essay_word_count

nm_essaycount=Normalizer()
nm_essaycount.fit(X_train.essay_word_count.values.reshape(-1,1))
X_train_essay_word_count_norm = nm_essaycount.transform(X_train['price'].values.reshape(1,
X_cv_essay_word_count_norm = nm_essaycount.transform(X_cv['price'].values.reshape(1,-1).T)
X_test_essay_word_count_norm = nm_essaycount.transform(X_test['price'].values.reshape(1,-1
```

```
print(X_train_essay_word_count_norm.shape,y_train.shape)
print(X_cv_essay_word_count_norm.shape,y_cv.shape)
print(X_test_essay_word_count_norm.shape,y_test.shape)
```

```
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

Vectorizing text data

BAg of Words

```
#train essays
#preprocessed_essays=X_train['essay'].values

print("before fitting")
print(X_train.shape,y_train.shape)
print(X_cv.shape,y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)

vector_essay=CountVectorizer(min_df=10,ngram_range=(1,4),max_features=5000)
vector_essay.fit(preprocessed_essays_train)
#we are fitting train essays only and we will transform the train, text and cv data bases


X_train_essay_bow=vector_essay.transform(preprocessed_essays_train)
X_train_cv_bow=vector_essay.transform(preprocessed_essays_cv)
X_test_bow=vector_essay.transform(preprocessed_essays_test)

print("="*100)
print("after fitting")
print(X_train_essay_bow.shape,y_train.shape)
print(X_train_cv_bow.shape,y_cv.shape)
print(X_test_bow.shape, y_test.shape)
```

```
before fitting
(49041, 21) (49041,)
(24155, 21) (24155,)
(36052, 21) (36052,)
=====================================================================================
=====================================================================================
after fitting
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
```

```
#titles


print("before fitting")
print(X_train.shape,y_train.shape)
print(X_cv.shape,y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
vector_title=CountVectorizer(min_df=10,tokenizer = lambda x: x.split(),max_features=5000)
vector_title.fit(preprocessed_titles_train)
#we are fitting train essays only and we will transform the train, text and cv data bases


X_train_title_bow=vector_title.transform(preprocessed_titles_train)
X_cv_title_cv_bow=vector_title.transform(preprocessed_titles_cv)
X_test_title_bow=vector_title.transform(preprocessed_titles_test)


print("="*100)
print("after fitting")
print(X_train_title_bow.shape,y_train.shape)
print(X_cv_title_cv_bow.shape,y_cv.shape)
print(X_test_title_bow.shape, y_test.shape)
```

```
before fitting
(49041, 21) (49041,)
(24155, 21) (24155,)
(36052, 21) (36052,)
=====================================================================================
=====================================================================================
after fitting
(49041, 2038) (49041,)
(24155, 2038) (24155,)
(36052, 2038) (36052,)
```

Tfidf

```
vec_tfidf_title=TfidfVectorizer(min_df=10,max_features=5000)
```

```
vec_tfidf_title=TfidfVectorizer(min_df=10,max_features=5000)
vec_tfidf_title.fit(preprocessed_titles_train)
X_train_title_tfidf=vec_tfidf_title.transform(preprocessed_titles_train)
X_train_title_cv_tfidf=vec_tfidf_title.transform(preprocessed_titles_cv)
X_test_title_tfidf=vec_tfidf_title.transform(preprocessed_titles_test)

print("="*100)
print("after fitting")
print(X_train_title_tfidf.shape,y_train.shape)
print(X_train_title_cv_tfidf.shape,y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
```

```
=====================================================================================
after fitting
(49041, 2038) (49041,)
(24155, 2038) (24155,)
(36052, 2038) (36052,)
```

```
vec_tfidf_essay=TfidfVectorizer(min_df=10,max_features=5000)
vec_tfidf_essay.fit(preprocessed_essays_train)
X_train_essay_tfidf=vec_tfidf_essay.transform(preprocessed_essays_train)
X_train_essay_cv_tfidf=vec_tfidf_essay.transform(preprocessed_essays_cv)
X_test_essay_tfidf=vec_tfidf_essay.transform(preprocessed_essays_test)

print("="*100)
print("after fitting")
print(X_train_essay_tfidf.shape,y_train.shape)
print(X_train_essay_cv_tfidf.shape,y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
```

```
=====================================================================================
after fitting
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
```

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pick
# make sure you have the glove_vectors file

import gdown
url = 'https://drive.google.com/uc?id=1MqUasf7jYoPbG35MJ28VQcOjjNp-ZDDp'
output = 'glove_vectors'
gdown.download(url, output, quiet=False)
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

```
Downloading...
From: https://drive.google.com/uc?id=1MqUasf7jYoPbG35MJ28VQcOjjNp-ZDDp
To: /content/glove_vectors
128MB [00:01, 68.8MB/s]
```

```
# average Word2Vec for train essays
# computing average word2vec for each review.
```

```
avg_w2v_train_essays=[]
for sentence in tqdm(preprocessed_essays_train):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if word in glove_words:
            cnt_wrds+=1
            vector+=model[word]
        if cnt_wrds:
            vector/=cnt_wrds

        avg_w2v_train_essays.append(vector)
```

```
100%|██████████| 49041/49041 [00:33<00:00, 1461.43it/s]
```

```
# average Word2Vec for cv essays
# computing average word2vec for each review.
avg_w2v_essays_cv=[]
for sentence in tqdm(preprocessed_essays_cv):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if word in glove_words:
            cnt_wrds+=1
            vector+=model[word]
        if cnt_wrds:
            vector/=cnt_wrds

        avg_w2v_essays_cv.append(vector)


# average Word2Vec for test essays
# computing average word2vec for each review.
avg_w2v_essays_test=[]
for sentence in tqdm(preprocessed_essays_test):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if word in glove_words:
            cnt_wrds+=1
            vector+=model[word]
        if cnt_wrds:
            vector/=cnt_wrds

        avg_w2v_essays_test.append(vector)


# average Word2Vec for train titles
# computing average word2vec for each review
# computing average word2vec for each review.
avg_w2v_titles_train=[]
for sentence in tqdm(preprocessed_titles_train):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if word in glove_words:
            cnt_wrds+=1
            vector+=model[word]
        if cnt_wrds:
            vector/=cnt_wrds

        avg_w2v_titles_train.append(vector)


# average Word2Vec for cv titles
# computing average word2vec for each review.
avg_w2v_titles_cv=[]
for sentence in tqdm(preprocessed_titles_cv):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if word in glove_words:
            cnt_wrds+=1
            vector+=model[word]
        if cnt_wrds:
            vector/=cnt_wrds

        avg_w2v_titles_cv.append(vector)


# average Word2Vec for test titles
# computing average word2vec for each review.
avg_w2v_titles_test=[]
for sentence in tqdm(preprocessed_titles_test):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if word in glove_words:
            cnt_wrds+=1
            vector+=model[word]
        if cnt_wrds:
            vector/=cnt_wrds

        avg_w2v_titles_test.append(vector)
```

weighted tfidf w2v

```
vec_tfidf_w2v=TfidfVectorizer()
t=vec_tfidf_w2v.fit(preprocessed_essays_train)
dictionary=dict(zip(vec_tfidf_w2v.get_feature_names(),list(vec_tfidf_w2v.idf_)))
tfidf_words=set(vec_tfidf_w2v.get_feature_names())


tfidf_w2v_essays_train=[]
```

```
tfidf_w2v_essays_train=[]
for sentence in tqdm(preprocessed_essays_train):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if (word in glove_words) and (word in tfidf_words):
            tf_idf=dictionary[word]*(sentence.count(word)/len(sentence.split(' ')))


            cnt_wrds+=tf_idf
            vector+=(model[word]*tf_idf)
        if cnt_wrds:
            vector/=cnt_wrds

        tfidf_w2v_essays_train.append(vector)



len(tfidf_w2v[0])


tfidf_w2v_essays_test=[]
for sentence in tqdm(preprocessed_essays_test):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if (word in glove_words) and (word in tfidf_words):
            tf_idf=dictionary[word]*(sentence.count(word)/len(sentence.split(' ')))


            cnt_wrds+=tf_idf
            vector+=(model[word]*tf_idf)
        if cnt_wrds:
            vector/=cnt_wrds

        tfidf_w2v_essays_test.append(vector)


tfidf_w2v_essays_cv=[]
for sentence in tqdm(preprocessed_essays_cv):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if (word in glove_words) and (word in tfidf_words):
            tf_idf=dictionary[word]*(sentence.count(word)/len(sentence.split(' ')))


            cnt_wrds+=tf_idf
            vector+=(model[word]*tf_idf)
        if cnt_wrds:
            vector/=cnt_wrds

        tfidf_w2v_essays_cv.append(vector)


vec_tdidf_w2v_titles=TfidfVectorizer()
t=vec_tdidf_w2v_titles.fit(preprocessed_titles_train)
dictionary=dict(zip(vec_tdidf_w2v_titles.get_feature_names(),list(vec_tdidf_w2v_titles.idf
dictionary=dict(zip(vec_tdidf_w2v_titles.get_feature_names(),list(vec_tdidf_w2v_titles.idf_
tfidf_words=set(vec_tdidf_w2v_titles.get_feature_names())


tfidf_title_train_w2v=[]
for sentence in tqdm(preprocessed_titles_train):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if (word in glove_words) and (word in tfidf_words):
            tf_idf=dictionary[word]*(sentence.count(word)/len(sentence.split(' ')))


            cnt_wrds+=tf_idf
            vector+=(model[word]*tf_idf)
        if cnt_wrds:
            vector/=cnt_wrds

        tfidf_title_train_w2v.append(vector)


tfidf_title_test_w2v=[]
for sentence in tqdm(preprocessed_titles_test):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if (word in glove_words) and (word in tfidf_words):
            tf_idf=dictionary[word]*(sentence.count(word)/len(sentence.split(' ')))


            cnt_wrds+=tf_idf
            vector+=(model[word]*tf_idf)
        if cnt_wrds:
            vector/=cnt_wrds

        tfidf_title_test_w2v.append(vector)


tfidf_title_cv_w2v=[]
for sentence in tqdm(preprocessed_titles_cv):
    vector=np.zeros(300)
    cnt_wrds=0
    for word in sentence.split(' '):
        if (word in glove_words) and (word in tfidf_words):
            tf_idf=dictionary[word]*(sentence.count(word)/len(sentence.split(' ')))


            cnt_wrds+=tf_idf
            vector+=(model[word]*tf_idf)
        if cnt_wrds:
            vector/=cnt_wrds

        tfidf_title_cv_w2v.append(vector)
```

Apply Multinomial NaiveBayes on these feature sets

categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)

```
from scipy.sparse import hstack


X_tr=hstack((X_train_state_ohe,X_train_teacher_ohe,X_train_grade_ohe,X_train_clean_subcate
X_train_essay_bow,X_train_title_bow)).tocsr()

X_cr=hstack((X_cv_state_ohe,X_cv_teacher_ohe,X_cv_grade_ohe,X_cv_clean_subcategories_ohe,X
X_train_cv_bow,X_cv_title_cv_bow)).tocsr()

X_te=hstack((X_test_state_ohe,X_test_teacher_ohe,X_test_grade_ohe,X_clean_subcategories_gr
X_test_bow,X_test_title_bow)).tocsr()


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

```
Final Data matrix
(49041, 7142) (49041,)
(24155, 7142) (24155,)
(36052, 7142) (36052,)
=====================================================================================
```

Gridsearch-cv using cv = 10

```
from sklearn.naive_bayes import MultinomialNB
nb=MultinomialNB(class_prior=[0.5,0.5])
paramaters={'alpha':[10**x for x in range(-4,4)]}
clf=GridSearchCV(nb,paramaters,cv=10,scoring='roc_auc',return_train_score=True,verbose=2,n
clf.fit(X_tr,y_train)
```

```
Fitting 10 folds for each of 8 candidates, totalling 80 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
[Parallel(n_jobs=-1)]: Done  37 tasks      | elapsed:    5.7s
[Parallel(n_jobs=-1)]: Done  80 out of  80 | elapsed:   10.1s finished
GridSearchCV(cv=10, error_score=nan,
             estimator=MultinomialNB(alpha=1.0, class_prior=[0.5, 0.5],
                                     fit_prior=True),
             iid='deprecated', n_jobs=-1,
             param_grid={'alpha': [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=2)
```

```
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']
```

```
clf.cv results
```

_ _

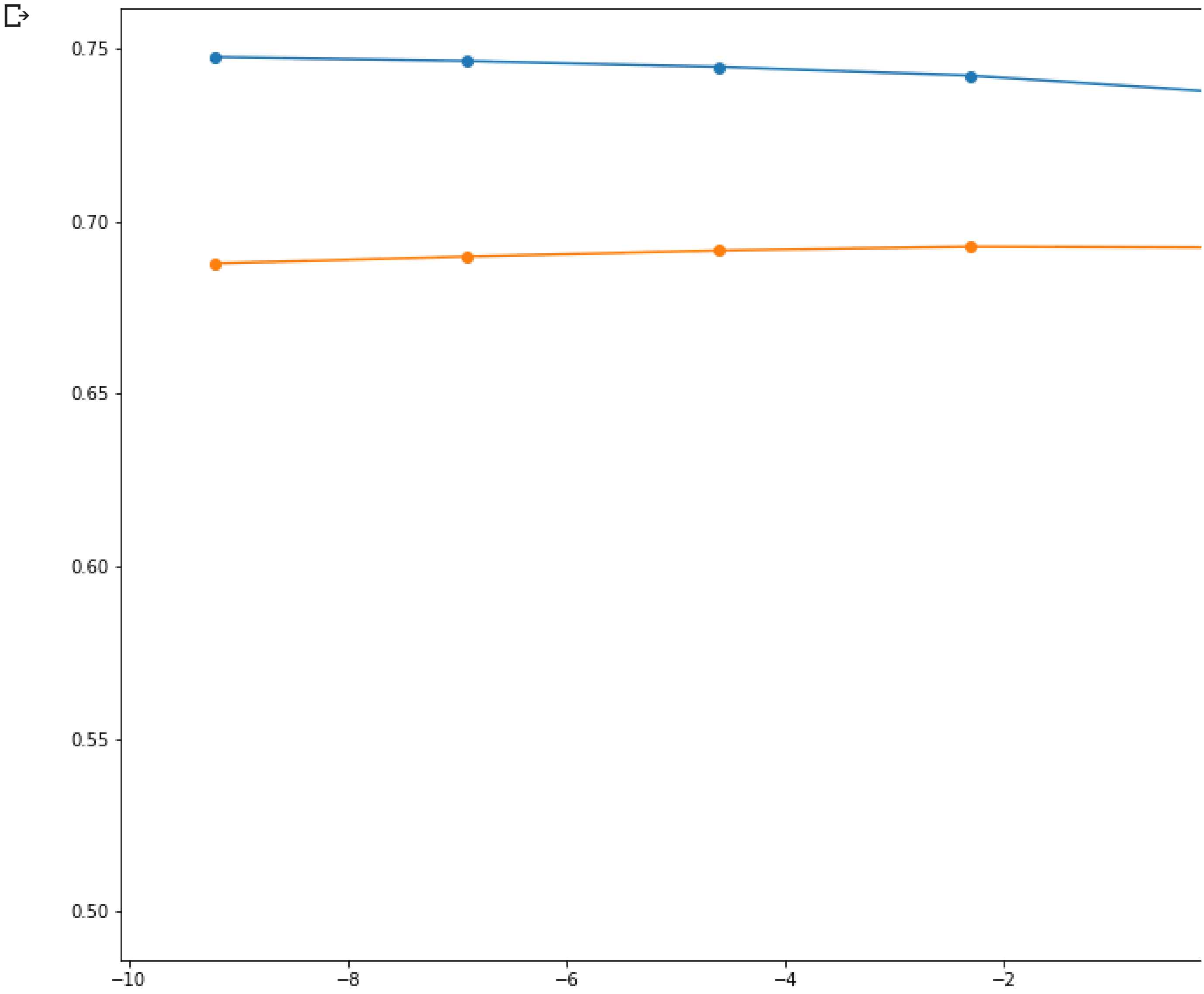
```
{'mean_fit_time': array([0.11186938, 0.10270519, 0.09645195, 0.09496608, 0.09696364,
        0.10031734, 0.11000118, 0.09886687]),
 'mean_score_time': array([0.01185446, 0.01150045, 0.01147881, 0.01441522, 0.013661
        0.01136398, 0.011183  , 0.0101053 ]),
 'mean_test_score': array([0.68774422, 0.68971348, 0.69146353, 0.69261183, 0.69235088
        0.68596742, 0.6295279 , 0.50011074]),
 'mean_train_score': array([0.74746862, 0.7463147 , 0.74461288, 0.74207194, 0.7372522
        0.72385648, 0.6478203 , 0.50011064]),
 'param_alpha': masked_array(data=[0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000],
```

```
alphas = [10**x for x in range(-4,4)]
log_alphas =[math.log(i) for i in alphas] #or use plt.axis('log')
plt.figure(figsize=(20,10))

plt.plot(log_alphas,train_auc,label="Train_AUC")

plt.plot(log_alphas,cv_auc,label='Test_AUC')


plt.scatter(log_alphas,train_auc,label="Train_AUC_points")
plt.scatter(log_alphas,cv_auc,label='Test_AUC_points')
plt.show()
```



```
        0.69451273, 0.63322064, 0.50134771]),
36000
```

```
             0.00030377  0.00022276  0.00140337])
```

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

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your tr_loop will be 49041 - 49041%1000 = 490
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    if data.shape[0]%1000 !=0:
        y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

```
best_alpha=clf.best_params_
best_alpha['alpha']
#Using bestparams attribute of gridsearch cv we can obtain the optimal value of alpha amon
#it simplifes our task and we can be rest assured that selected hyperparameter is most opt
```

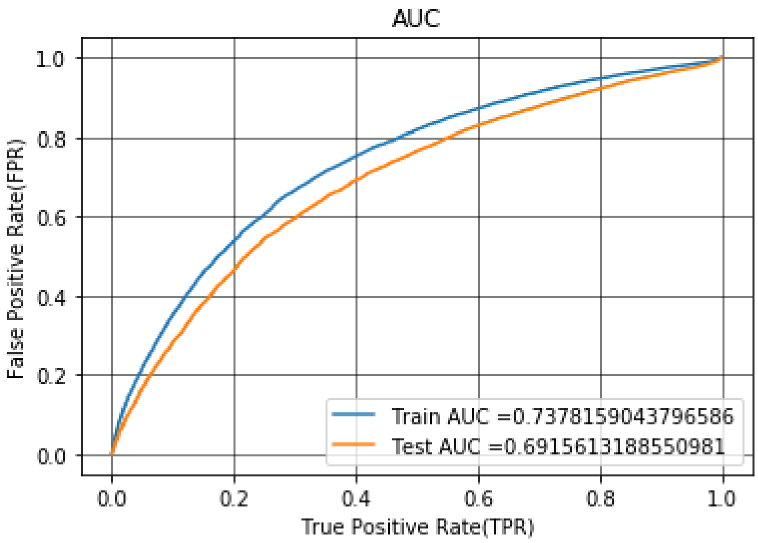
```
0.1
```

```
from sklearn.metrics import roc_curve


nb_bow=MultinomialNB(alpha=0.1,class_prior=[0.5,0.5])
nb_bow.fit(X_tr,y_train)
y_train_pred = batch_predict(nb_bow, X_tr)
y_test_pred = batch_predict(nb_bow, X_te)

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


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



For Bow model for alpha=0.1 ,we get train AUC of 0.744 and Test AUC of 0.6898 this is better than

printing the confusion matrix with predicted and original labe

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

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

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

```
print("train confusion matrix")
print(confusion_matrix(y_train,predict(y_train_pred,tr_thresholds,train_fpr,train_tpr)))
```

```
train confusion matrix
the maximum value of tpr*(1-fpr) 0.4671703561487076 for threshold 0.532
[[ 5018  2408]
 [12982 28633]]
```

```
print("test confusion matrix")
print(confusion_matrix(y_test,predict(y_test_pred,te_thresholds,test_fpr, test_tpr)))
```

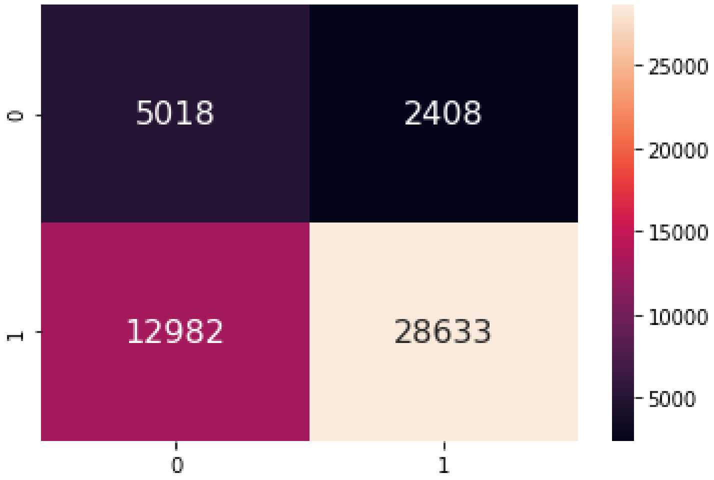
```
    test confusion matrix
```

```
conf_matr_df_train_1=pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred,tr_thresho
```

```
the maximum value of tpr*(1-fpr) 0.4671703561487076 for threshold 0.532
```

```
sns.heatmap(conf_matr_df_train_1,annot=True, annot_kws={"size":16},fmt='g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff68437dac8>
```



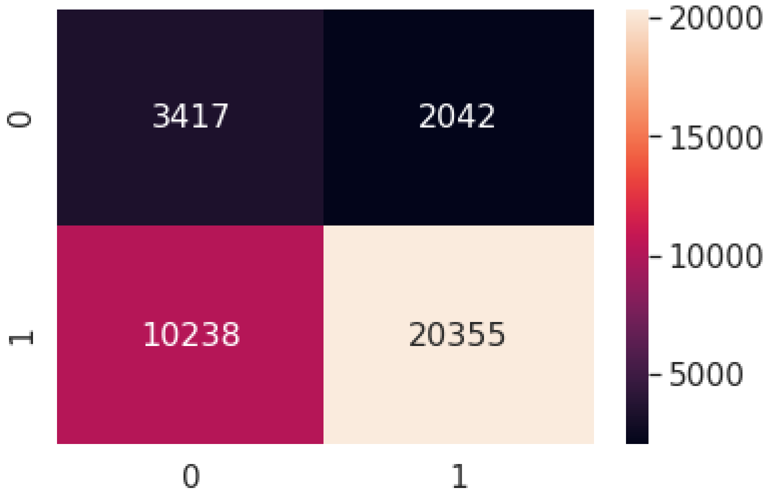
```
conf_matr_df_test_1=pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred,te_thresholds
```

```
the maximum value of tpr*(1-fpr) 0.4221775437724126 for threshold 0.57
```

```
sns.set_style('whitegrid')
sns.set(font_scale=1.4)
```

```
sns.heatmap(conf_matr_df_test_1,annot=True,annot_kws={"size":16},fmt='g')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff66e4afac8>
```



categorical, numerical features + project_title(TFIDF) + prepr

```
X_tr1=hstack((X_train_state_ohe,X_train_teacher_ohe,X_train_grade_ohe,X_train_clean_subcat
X_train_essay_tfidf,X_train_title_tfidf)).tocsr()

X_cr1=hstack((X_cv_state_ohe,X_cv_teacher_ohe,X_cv_grade_ohe,X_cv_clean_subcategories_ohe,
X_train_essay_cv_tfidf,X_train_title_cv_tfidf)).tocsr()

X_te1=hstack((X_test_state_ohe,X_test_teacher_ohe,X_test_grade_ohe,X_clean_subcategories_g
X_test_essay_tfidf,X_test_title_tfidf)).tocsr()


print("Final Data matrix")
print(X_tr1.shape, y_train.shape)
print(X_cr1.shape, y_cv.shape)
print(X_te1.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 7142) (49041,)
(24155, 7142) (24155,)
(36052, 7142) (36052,)
====================================================================================
```

```
from sklearn.naive_bayes import MultinomialNB
nb=MultinomialNB(class_prior=[0.5,0.5])
paramaters={'alpha':[0.00001, 0.0001,0.001, 0.01, 0.1,0.25,0.5,0.8, 1, 100]}
clf=GridSearchCV(nb,paramaters,cv=10,scoring='roc_auc',return_train_score=True,verbose=2)
clf.fit(X_tr1,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']
```

```
clf.cv_results_
```

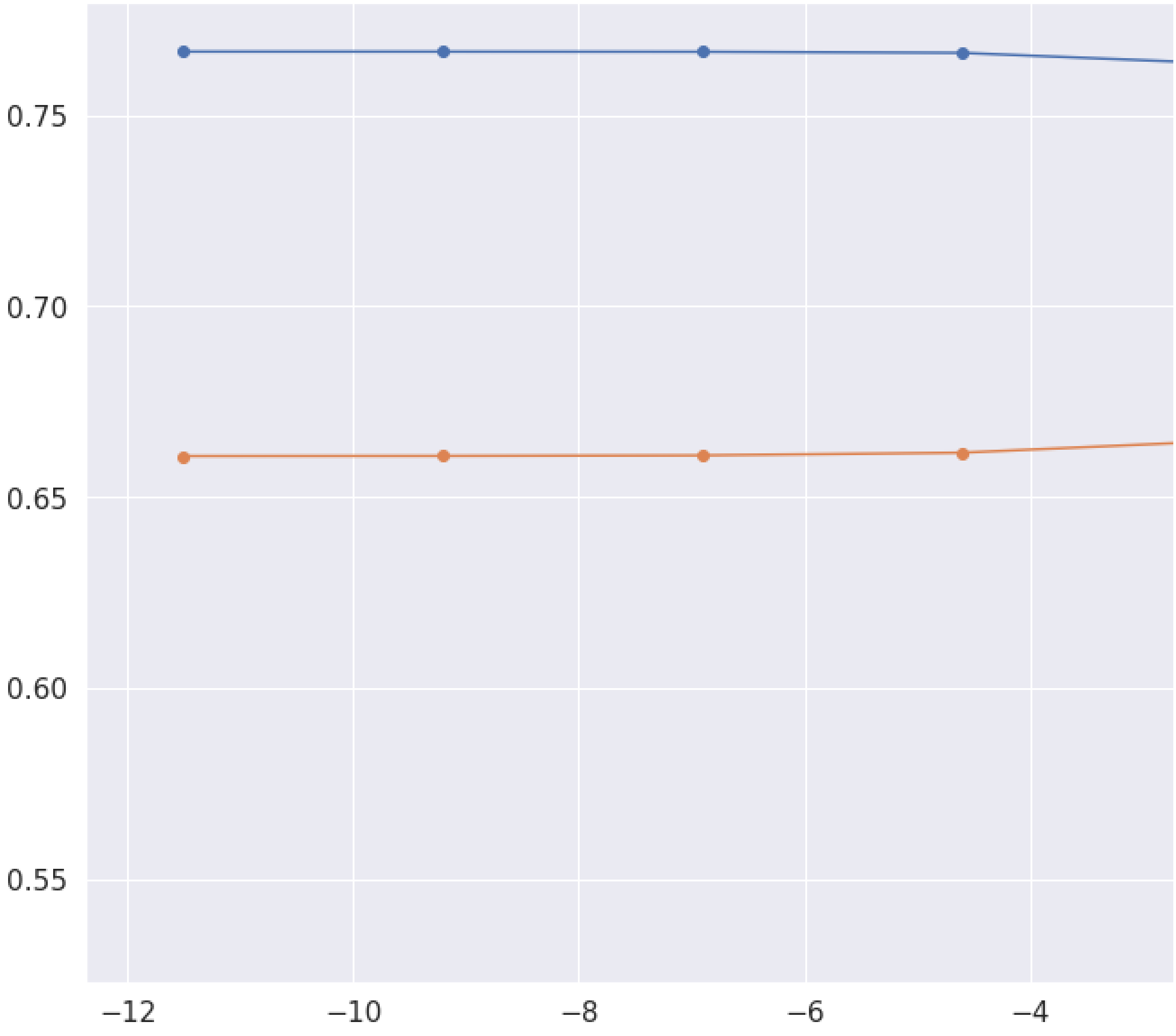
```
{'mean_fit_time': array([0.06001227, 0.06171432, 0.05910056, 0.05969627, 0.06015861,
        0.05987263, 0.05930195, 0.05854492, 0.0587913 , 0.05843284]),
 'mean_score_time': array([0.00695708, 0.00714757, 0.00647652, 0.00670424, 0.00669894
        0.00656168, 0.00659227, 0.00645094, 0.00656333, 0.00639853]),
 'mean_test_score': array([0.66068614, 0.6607264 , 0.66087544, 0.66157048, 0.66463586
        0.66626703, 0.6654091 , 0.66198552, 0.65899958, 0.53584202]),
 'mean_train_score': array([0.76664544, 0.76663501, 0.76659712, 0.76631703, 0.7635352
        0.75872747, 0.75048957, 0.74059843, 0.73414247, 0.54153761]),
 'param_alpha': masked_array(data=[1e-05, 0.0001, 0.001, 0.01, 0.1, 0.25, 0.5, 0.8, 1
                    100],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
             dtype=object),
 'params': [{'alpha': 1e-05},
  {'alpha': 0.0001},
  {'alpha': 0.001},
  {'alpha': 0.01},
  {'alpha': 0.1},
  {'alpha': 0.25},
  {'alpha': 0.5},
  {'alpha': 0.8},
  {'alpha': 1},
  {'alpha': 100}],
 'rank_test_score': array([ 8,  7,  6,  5,  3,  1,  2,  4,  9, 10], dtype=int32),
 'split0_test_score': array([0.65767894, 0.65772745, 0.65788461, 0.65864843, 0.661561
        0.66267609, 0.66130465, 0.65741992, 0.65432488, 0.53747939]),
 'split0_train_score': array([0.7668678 , 0.76685837, 0.76681912, 0.76651233, 0.76367
        0.75882131, 0.75048328, 0.74046659, 0.73393443, 0.54136759]),
 'split1_test_score': array([0.65018503, 0.65019797, 0.65024228, 0.65057609, 0.652282
        0.65247444, 0.65037134, 0.64629517, 0.64320132, 0.53263221]),
 'split1_train_score': array([0.76692916, 0.7669193 , 0.76688176, 0.76660975, 0.76387
        0.75913929, 0.75102913, 0.74130844, 0.73495708, 0.54203277]),
 'split2_test_score': array([0.66529716, 0.6653211 , 0.66541943, 0.66608186, 0.669111
        0.67032494, 0.66871478, 0.6646027 , 0.66122292, 0.53435655]),
 'split2_train_score': array([0.76550301, 0.76549212, 0.76545392, 0.76517397, 0.76236
        0.75747506, 0.74914733, 0.73917177, 0.73268578, 0.54154061]),
 'split3_test_score': array([0.66490934, 0.66491645, 0.66498664, 0.66558115, 0.669079
        0.6707367 , 0.6696887 , 0.66605081, 0.66296182, 0.54112549]),
 'split3_train_score': array([0.76497669, 0.76495898, 0.76490654, 0.76459685, 0.76166
        0.75669324, 0.7483191 , 0.73834981, 0.73187727, 0.54078396]),
 'split4_test_score': array([0.64698962, 0.64703879, 0.64723254, 0.64814468, 0.652904
        0.65661984, 0.65820089, 0.65673046, 0.65461054, 0.5460213 ]),
 'split4_train_score': array([0.76849989, 0.76849103, 0.76845749, 0.76818887, 0.76542
        0.76059565, 0.75229214, 0.74228795, 0.73577532, 0.54049038]),
 'split5_test_score': array([0.67477179, 0.6747912 , 0.67486527, 0.67549633, 0.679039
        0.68120434, 0.68064185, 0.67712752, 0.67376941, 0.52418164]),
 'split5_train_score': array([0.76608201, 0.76607332, 0.76603814, 0.76576601, 0.763
        0.75821464, 0.74998415, 0.74010057, 0.73362925, 0.54274411]),
 'split6_test_score': array([0.63932078, 0.63936903, 0.63949338, 0.64013129, 0.643495
        0.64583687, 0.64615712, 0.64379037, 0.64145309, 0.53530628]),
 'split6_train_score': array([0.76679776, 0.76678703, 0.76674839, 0.76646279, 0.76362
        0.7587638 , 0.75053503, 0.74073032, 0.73436774, 0.54226667]),
 'split7_test_score': array([0.66380006, 0.66390692, 0.66435896, 0.66564191, 0.667853
        0.66910249, 0.66823435, 0.66462157, 0.66157935, 0.53249818]),
 'split7_train_score': array([0.76713991, 0.7671317 , 0.76709851, 0.76682617, 0.76403
        0.75924585, 0.75105555, 0.74125082, 0.73484723, 0.54175366]),
 'split8_test_score': array([0.66426732, 0.66429031, 0.66435864, 0.66469832, 0.667397
        0.66874306, 0.66738726, 0.66417212, 0.66149548, 0.53591052]),
 'split8_train_score': array([0.76711545, 0.76710592, 0.76707095, 0.76680672, 0.76413
        0.75945729, 0.75133423, 0.74148149, 0.7350065 , 0.54162179]),
 'split9_test_score': array([0.67964131, 0.67970477, 0.67991266, 0.68070471, 0.683633
        0.68495151, 0.68339009, 0.67904452, 0.67537702, 0.5389087 ]),
 'split9_train_score': array([0.76654271, 0.76653233, 0.76649639, 0.76622685, 0.76355
        0.75886855, 0.75071572, 0.74083653, 0.73434413, 0.54077456]),
 'std_fit_time': array([0.00202863, 0.00521401, 0.00086507, 0.00075687, 0.00118948,
        0.00211662, 0.00073502, 0.00172722, 0.00154177, 0.00074669]),
 'std_score_time': array([0.00058048, 0.00095907, 0.00015293, 0.00072195, 0.00021571,
        0.00018258, 0.00024313, 0.00020482, 0.00028561, 0.00013615]),
 'std_test_score': array([0.01175888, 0.0117593 , 0.01176979, 0.01179895, 0.01171647,
        0.01156044, 0.01123418, 0.01086787, 0.0105758 , 0.00550968]),
 'std_train_score': array([0.00092062, 0.00092241, 0.00092616, 0.00093409, 0.00097376
        0.00102278, 0.00106749, 0.00109522, 0.00110563, 0.00067542])}
```

```
alphas = [0.00001, 0.0001,0.001, 0.01, 0.1,0.25,0.5,0.8, 1, 100]
log_alphas =[math.log(i) for i in alphas]
plt.figure(figsize=(20,10))

plt.plot(log_alphas,train_auc,label="Train_AUC")


plt.plot(log_alphas,cv_auc,label='Test_AUC')


plt.scatter(log_alphas,train_auc,label="Train_AUC_points")
plt.scatter(log_alphas,cv_auc,label='Test_AUC_points')
plt.show()
```



```
best_aplha=clf.best_params_
best_aplha
```

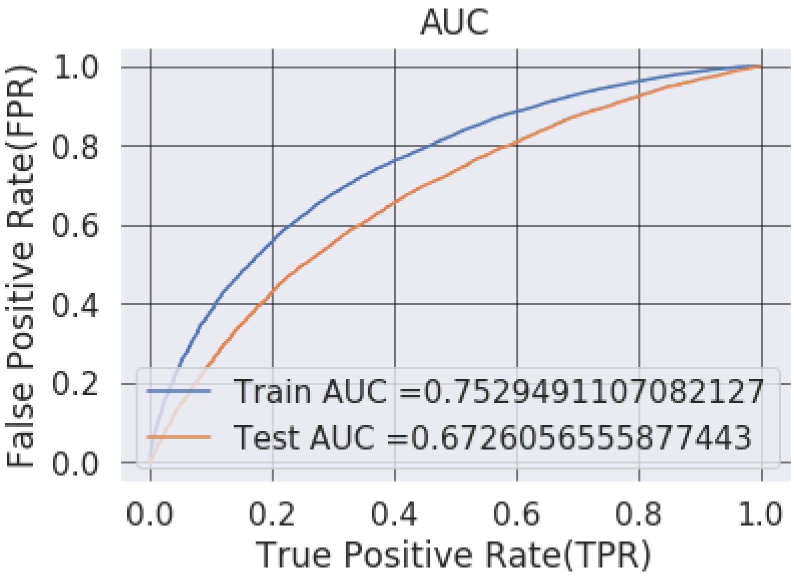
```
{'alpha': 0.25}
```

```
from sklearn.metrics import roc_curve

nb_tfidf=MultinomialNB(alpha=0.25,class_prior=[0.5,0.5])
nb_tfidf.fit(X_tr1,y_train)
y_train_pred = batch_predict(nb_tfidf, X_tr1)
y_test_pred = batch_predict(nb_tfidf, X_te1)

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()
```



From given plot we observe that at alpha=0.25 we get train AUC of 0.740 and test AUC of 0.696

```
print("train confusion matrix")
print(confusion_matrix(y_train,predict(y_train_pred,tr_thresholds,train_fpr,train_tpr)))
```

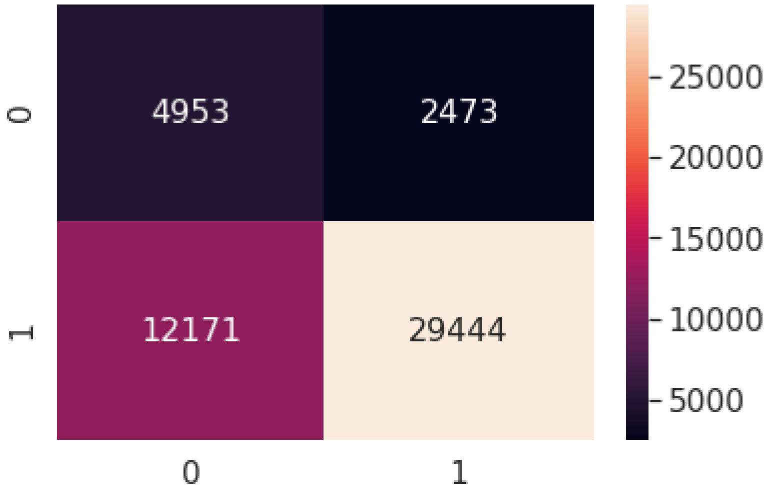
```
train confusion matrix
the maximum value of tpr*(1-fpr) 0.4754070916506357 for threshold 0.481
[[ 4953  2473]
 [12171 29444]]
```

```
print("test confusion matrix")
print(confusion_matrix(y_test,predict(y_test_pred,te_thresholds,test_fpr,test_tpr)))
```

```
test confusion matrix
the maximum value of tpr*(1-fpr) 0.3946630153108321 for threshold 0.515
[[ 3384  2075]
 [11206 19387]]
```

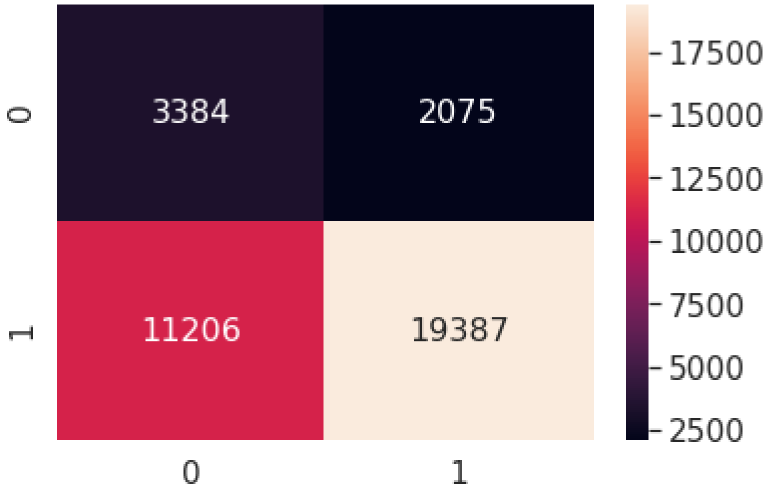
```
conf_matr_df_train_1=pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred,tr_thresho
sns.heatmap(conf_matr_df_train_1,annot=True, annot_kws={"size":16},fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.4754070916506357 for threshold 0.481
<matplotlib.axes._subplots.AxesSubplot at 0x7ff641a63be0>
```



```
conf_matr_df_test_1=pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred,te_thresholds
sns.heatmap(conf_matr_df_test_1,annot=True,annot_kws={"size":16},fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.3946630153108321 for threshold 0.515
<matplotlib.axes._subplots.AxesSubplot at 0x7ff64acdbef0>
```



Select best 30 features of both Positive and negative class f

```
from scipy.sparse import csr_matrix
X11_tr=X_tr
X11_cv=X_cv
X11_te=X_te
print("The final Data Matrix for Set:1" , " All the shapes of the data represent the merge
print("shape of X_train is : ",            X11_tr.shape)
print("shape of X_Cross validation is :" , X11_cv.shape)
print("shape of X_test is ",               X11_te.shape)
```

```
The final Data Matrix for Set:1  All the shapes of the data represent the merged feat
shape of X_train is :  (49041, 7142)
shape of X_Cross validation is : (24155, 21)
shape of X_test is  (36052, 7142)
```

```
nb_bow = MultinomialNB(alpha =0.1 ,class_prior=[0.5,0.5])
nb_bow.fit(X1_tr, y_train)
```

```
MultinomialNB(alpha=0.1, class_prior=[0.5, 0.5], fit_prior=True)
```

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

```
rflow.com/questions/14131615/possible-to-append-multiple-lists-at-once-python
ort chain
st = list(chain(vec_state.get_feature_names(),vec_prefix.get_feature_names(),vec_grade.get_
                vec_cat.get_feature_names(),vec_sub.get_feature_names(),vector_essay.get_fe
                vector_title.get_feature_names(),X_train_price_norm,X_train_No_of_teachers_
```

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

```
The words with highest importance in Postive class are
['would benefit' 'new experiences' 'make ends' 'want give students'
 'year olds' 'class nannan' 'use computers' 'materials help students'
 'day day' 'skills necessary' 'able control home' 'reading grade'
 'come different backgrounds' '10' 'love coming school' 'national'
 'work hard every' 'need classroom' 'not access technology'
 'many children' '100 free' '100 percent' '000' '100' 'help children'
 'learn day' 'classroom come' 'learning classroom' 'school college']
********************
The words with highest importance in Negative class are
['new experiences' 'year olds' 'books help' 'skills necessary' 'also help'
 'technology engineering math' 'would benefit' 'class nannan'
 'come different backgrounds' 'able control home' 'day day'
 'love coming school' '10' 'national' 'use computers' 'reading grade'
 'need classroom' 'work hard every' 'not access technology'
 'many children' '100 percent' '100 free' '100' '000' 'help children'
 'learn day' 'classroom come' 'learning classroom' 'school college']
```

```
#for set2 tfidf
```

```
from scipy.sparse import csr_matrix
X2_tr=X1_tr
X2_cv=X1_cv
X2_te=X1_te
print("The final Data Matrix for Set:1" , " All the shapes of the data represent the merge
print("shape of X_train is : ",            X2_tr.shape)
print("shape of X_Cross validation is :" , X2_cv.shape)
print("shape of X_test is ",               X2_te.shape)
print("shape of X_test is ",               X2_te.shape)
```

```
The final Data Matrix for Set:1  All the shapes of the data represent the merged feat
shape of X_train is :  (49041, 7142)
shape of X_Cross validation is : (24155, 21)
shape of X_test is  (36052, 7142)
```

```
nb_tf = MultinomialNB(alpha =0.25 ,class_prior=[0.5,0.5])
nb_tf.fit(X2_tr, y_train)
```

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

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

```
# https://stackoverflow.com/questions/14131615/possible-to-append-multiple-lists-at-once-p
from itertools import chain
Stacked_Feature_list = list(chain(vec_state.get_feature_names(),vec_prefix.get_feature_nam
                                  vec_cat.get_feature_names(),vec_sub.get_feature_names(),
                                  vec_tfidf_title.get_feature_names(),X_train_price_norm,X
```

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

```
The words with highest importance in Postive class are
['workbooks' 'motivating' 'length' 'videos' 'writing' 'canvas'
 'unfortunate' 'looked' 'controlled' 'satisfy' 'academically' 'primarily'
 'choices' '000' 'knowledge' 'mindset' 'wider' 'mo' 'neat' 'limitations'
 '100' '1000' '00' '10' 'fruit' 'impress' 'cares' 'informed' 'recess']
********************
The words with highest importance in Negative class are
['motivating' 'writing' 'behaviors' 'satisfy' 'alternative' 'tests'
 'workbooks' 'canvas' 'choices' 'academically' 'controlled' 'knowledge'
 '000' 'mindset' 'unfortunate' 'primarily' 'mo' 'wider' 'neat'
 'limitations' '1000' '100' '10' '00' 'fruit' 'impress' 'cares' 'informed'
 'recess']
```

```
#!pip install prettytable
```

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

from prettytable import PrettyTable



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

x.add_row(["BOW", "MultiNomialNaiveBayes", 0.1, 0.691])
x.add_row(["TFIDF", "MultiNomialNaiveBayes", 0.25, 0.672])

print(x)
```

```
+------------+-------------+-----------------------+------------+
| Vectorizer |    Model    | Alpha:Hyper Parameter |  Test AUC  |
+------------+-------------+-----------------------+------------+
|    BOW     | Naive Bayes |          0.1          |   0.6898   |
|   TFIDF    | Naive Bayes |          0.25         |   0.696    |
+------------+-------------+-----------------------+------------+
```

from the confusion matrix we can infer that the no of misclassified points are reasonably high. this use avgw2v or tfidfw2v to further enhance the performance of the models.

as for bow and tfidf on above features we are getting quite similar test auc.

```
jupyter nbconvert --to html notebook.ipynb
```

```
  File "<ipython-input-146-991260e3a7ca>", line 1
    jupyter nbconvert --to html notebook.ipynb
                    ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW