Table of Contents

Author : SAURABH SINGHAI

DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

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

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*

Feature	Description
project_essay_1	First application essay
project_essay_2	Second application essay*
project_essay_3	Third application essay*
project_essay_4	Fourth application essay*
project_submitted_datetime	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
teacher_id	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
teacher_prefix	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
id	A `project_id` value from the `train.csv` file. **Example:** `p036502`
description	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
quantity	Quantity of the resource required. **Example:** `3`
price	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

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

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

Import Libraries

In [1]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

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

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

from tqdm import tqdm
import os
import shutil, os

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

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.
```

Reading Data

In [2]:

```
#copy files before starting
import shutil, os

#shutil.copy('D:\_AI-ML\Assignments_DonorsChoose_2018\train_data.csv',
'C:\Users\ssinghai\Downloads')

#shutil.copy("D:\_AI-ML\Assignments_DonorsChoose_2018\resources.csv",
'C:\Users\ssinghai\Downloads')

project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna(' ')
```

Approved And Non Approved Projects

In [3]:

In [3]:

```
y_value_counts = project_data['project_is_approved'].value_counts()
print("Number of projects thar are approved for funding ", y_value_counts[1], ", (",
(y_value_counts[1]/(y_value_counts[1]+y_value_counts[0]))*100,"%)")
print("Number of projects thar are not approved for funding ", y_value_counts[0], ", (",
(y_value_counts[0]/(y_value_counts[1]+y_value_counts[0]))*100,"%)")
```

```
Number of projects thar are approved for funding  92706 , ( 84.8583040422 %)
Number of projects thar are not approved for funding  16542 , ( 15.1416959578 %)
```

Project Dataframe shape and Column Values

In [4]:

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

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

Preprocessing of `project_subject_categories`

In [5]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

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

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

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

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

Preprocessing of `project_subject_subcategories`

In [6]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python:
https://stackoverflow.com/a/47301924/4084039

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

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

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

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

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

Preprocessing of `project_grade_category`

In [7]:

```
print(project_data["project_grade_category"].values[0:10])
```

```
['Grades PreK-2' 'Grades 6-8' 'Grades 6-8' 'Grades PreK-2' 'Grades PreK-2'
 'Grades 3-5' 'Grades 6-8' 'Grades 3-5' 'Grades PreK-2' 'Grades PreK-2']
```

In [8]:

```
project_data["project_grade_category"] = project_data["project_grade_category"].str.replace(" ", "_
")
project_data["project_grade_category"] = project_data["project_grade_category"].str.replace("-", "_
")

print(project_data["project_grade_category"].values[0:10])
```

```
['Grades_PreK_2' 'Grades_6_8' 'Grades_6_8' 'Grades_PreK_2' 'Grades_PreK_2'
 'Grades_3_5' 'Grades_6_8' 'Grades_3_5' 'Grades_PreK_2' 'Grades_PreK_2']
```

Create new column 'Essay' by merging all project Essays

In [9]:

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

In [10]:

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

Out[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [11]:

```
# printing some random reviews
print(project_data['essay'].values[0])
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner Teacher and will be sent home regularly to watch.  The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd's for the years to come for other EL students.\r\nnannan
```

Use Decontraction function to decontract project essay

In [12]:

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

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

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

In [13]:

```
sent = decontracted(project_data['essay'].values[0])
print(sent)
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner is have a strong support system at home t
hat begs for more resources.  Many times our parents are learning to read and speak English along
side of their children.  Sometimes this creates barriers for parents to be able to help their
child learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd
is and players, students are able to continue their mastery of the English language even if no one
at home is able to assist.  All families with students within the Level 1 proficiency status, will
be a offered to be a part of this program.  These educational videos will be specially chosen by t
he English Learner Teacher and will be sent home regularly to watch.  The videos are to help the c
hild develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will hav
e the opportunity to check out a dvd player to use for the year.  The plan is to use these videos
and educational dvd is for the years to come for other EL students.\r\nnannan
==================================================
```

Remove line breaks

In [14]:

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

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school.      We have over 24 languages represented in our English Learner program with
students at every level of mastery.  We also have over 40 countries represented with the families
within our school.  Each student brings a wealth of knowledge and experiences to us that open our
eyes to new cultures, beliefs, and respect. The limits of your language are the limits of your wor
ld. -Ludwig Wittgenstein  Our English learner is have a strong support system at home that begs fo
r more resources.  Many times our parents are learning to read and speak English along side of
their children.  Sometimes this creates barriers for parents to be able to help their child learn
phonetics, letter recognition, and other reading skills.    By providing these dvd is and players,
students are able to continue their mastery of the English language even if no one at home is able
to assist.  All families with students within the Level 1 proficiency status, will be a offered to
be a part of this program.  These educational videos will be specially chosen by the English Learn
er Teacher and will be sent home regularly to watch.  The videos are to help the child develop ear
ly reading skills.    Parents that do not have access to a dvd player will have the opportunity to
check out a dvd player to use for the year.  The plan is to use these videos and educational dvd i
s for the years to come for other EL students.  nannan
```

Remove Special Chars

In [15]:

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

```
My students are English learners that are working on English as their second or third languages We
are a melting pot of refugees immigrants and native born Americans bringing the gift of language t
o our school We have over 24 languages represented in our English Learner program with students at
every level of mastery We also have over 40 countries represented with the families within our sch
ool Each student brings a wealth of knowledge and experiences to us that open our eyes to new cult
ures beliefs and respect The limits of your language are the limits of your world Ludwig Wittgenst
ein Our English learner is have a strong support system at home that begs for more resources Many
times our parents are learning to read and speak English along side of their children Sometimes th
times our parents are learning to read and speak English along side of their children Sometimes th
is creates barriers for parents to be able to help their child learn phonetics letter recognition
and other reading skills By providing these dvd is and players students are able to continue their
mastery of the English language even if no one at home is able to assist All families with student
s within the Level 1 proficiency status will be a offered to be a part of this program These
educational videos will be specially chosen by the English Learner Teacher and will be sent home r
egularly to watch The videos are to help the child develop early reading skills Parents that do no
t have access to a dvd player will have the opportunity to check out a dvd player to use for the y
ear The plan is to use these videos and educational dvd is for the years to come for other EL stud
ents nannan
```

Remove Stopwards and Join the essays

In [16]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "do
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [17]:

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

In [18]:

```
# after preprocesing
preprocessed_essays=Text_cleaner(project_data['essay'])
```

```
100%|██████████████████████████████████████████████████| 109248/109248
[01:16<00:00, 1420.12it/s]
```

In [19]:

```
preprocessed_essays[1]
```

Out[19]:

```
'students arrive school eager learn polite generous strive best know education succeed life help i
mprove lives school focuses families low incomes tries give student education deserve not much stu
dents use materials given best projector need school crucial academic improvement students technol
ogy continues grow many resources internet teachers use growth students however school limited res
ources particularly technology without disadvantage one things could really help classrooms
projector projector not crucial instruction also growth students projector show presentations docu
mentaries photos historical land sites math problems much projector make teaching learning easier
also targeting different types learners classrooms auditory visual kinesthetic etc nannan'
```

Remove Stopwards and Join the essays

In [20]:

```
project_data['essay'] = preprocessed_essays
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
project_data.head(5)
```

Out[20]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Gra
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Gra
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Gra

Preprocessing of `project_title`

Preprocessing of `project_title`

In [21]:

```
# similarly you can preprocess the titles also
preprocessed_project_title=Text_cleaner(project_data['project_title'])
```

```
100%|████████████████████████████████████████████████████████████████████████| 109248/109248
[00:03<00:00, 31566.58it/s]
```

In [22]:

```
preprocessed_project_title[1]
```

Out[22]:

```
'wanted projector hungry learners'
```

Drop column project_title and use Cleaned_Title

In [23]:

```
project_data['Cleaned_title']=preprocessed_project_title
project_data.drop(['project_title'], axis=1, inplace=True)
```

Add up the price based on project id

In [24]:

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

In [25]:

```
project_data.columns
```

Out[25]:

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

In [26]:

```
project_data.drop(['project_resource_summary'], axis=1, inplace=True)
project_data.drop(['Unnamed: 0'], axis=1, inplace=True)
project_data.drop(['id'], axis=1, inplace=True)
project_data.drop(['teacher_id'], axis=1, inplace=True)
```

Adding the word count of essay and title as new columns

In [27]:

```
project_data['essay_count']=project_data['essay'].str.len()
project_data['title_count']=project_data['Cleaned_title'].str.len()
```

we are going to consider

```
- school_state : categorical data
- clean_categories : categorical data
```

```
        - clean_subcategories : categorical data
        - project_grade_category : categorical data
        - teacher_prefix : categorical data

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

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

Separate out the Dependant and independant variables

In [28]:

```
#https://stackoverflow.com/questions/29763620/how-to-select-all-columns-except-one-column-in-panda
s
X=project_data.loc[:, project_data.columns != 'project_is_approved']
y=project_data['project_is_approved']
X.shape
```

Out[28]:

```
(109248, 13)
```

Splitting data into Test,Train,CV with Startified Sampling

In [29]:

```
#Splitting data into Test,Train,CV
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

from sklearn.model_selection import train_test_split

X_train,X_test,y_train, y_test=train_test_split(X, y, test_size=0.3, random_state=0)
X_train,X_cv,y_train,y_cv=train_test_split(X_train, y_train, test_size=0.3, random_state=0)


print(X_train.shape)
print(X_test.shape)
print(X_cv.shape)
print(y_train.shape)
print(y_test.shape)
print(y_cv.shape)
```

```
(53531, 13)
(32775, 13)
(22942, 13)
(53531,)
(32775,)
(22942,)
```

Vectorize the features

Vectorize the Categorical Features - categories

In [30]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer1 = CountVectorizer(lowercase=False, binary=True)
vectorizer1.fit(X_train['clean_categories'].values)
feature_names_bow=[]
feature_names_tfidf=[]
```

```
# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_categories=vectorizer1.transform(X_train['clean_categories'].values)
X_test_clean_categories=vectorizer1.transform(X_test['clean_categories'].values)
X_cv_clean_categories=vectorizer1.transform(X_cv['clean_categories'].values)


print(vectorizer1.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_clean_categories.shape)
feature_names_bow.append(vectorizer1.get_feature_names())
feature_names_tfidf.append(vectorizer1.get_feature_names())
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of matrix after one hot encodig  (53531, 9)
```

Vectorize the Categorical Features - subcategories

In [31]:

```
vectorizer2 = CountVectorizer(lowercase=False, binary=True)
vectorizer2.fit(X_train['clean_subcategories'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_sub_categories=vectorizer2.transform(X_train['clean_subcategories'].values)
X_test_clean_sub_categories=vectorizer2.transform(X_test['clean_subcategories'].values)
X_cv_clean_sub_categories=vectorizer2.transform(X_cv['clean_subcategories'].values)


print(vectorizer2.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_clean_sub_categories.shape)
feature_names_bow.extend(vectorizer2.get_feature_names())
feature_names_tfidf.append(vectorizer2.get_feature_names())
```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government',
'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics',
'EnvironmentalScience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness',
'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writing', 'M
athematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'Socia
lSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
Shape of matrix after one hot encodig  (53531, 30)
```

Vectorize the Categorical Features - school state

In [32]:

```
vectorizer3 = CountVectorizer(lowercase=False, binary=True)
vectorizer3.fit(X_train['school_state'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_skl_state=vectorizer3.transform(X_train['school_state'].values)
X_test_skl_state=vectorizer3.transform(X_test['school_state'].values)
X_cv_skl_state=vectorizer3.transform(X_cv['school_state'].values)


print(vectorizer3.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_skl_state.shape)
feature_names_bow.extend(vectorizer3.get_feature_names())
feature_names_tfidf.append(vectorizer3.get_feature_names())
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
Shape of matrix after one hot encodig  (53531, 51)
```

Vectorize the Categorical Features - teacher prefix

In [33]:

```
vectorizer4 = CountVectorizer(lowercase=False, binary=True)
vectorizer4.fit(X_train['teacher_prefix'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_prefix=vectorizer4.transform(X_train['teacher_prefix'].values)
X_test_teacher_prefix=vectorizer4.transform(X_test['teacher_prefix'].values)
X_cv_teacher_prefix=vectorizer4.transform(X_cv['teacher_prefix'].values)


print(vectorizer4.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_teacher_prefix.shape)
feature_names_bow.extend(vectorizer4.get_feature_names())
feature_names_tfidf.append(vectorizer4.get_feature_names())
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig  (53531, 5)
```

Vectorize the Categorical Features - project_grade_category

In [34]:

```
vectorizer5 = CountVectorizer(lowercase=False, binary=True)
vectorizer5.fit(X_train['project_grade_category'].values)

# we use the fitted CountVectorizer to convert the text to vector
X_train_project_grade_category=vectorizer5.transform(X_train['project_grade_category'].values)
X_test_project_grade_category=vectorizer5.transform(X_test['project_grade_category'].values)
X_cv_project_grade_category=vectorizer5.transform(X_cv['project_grade_category'].values)


print(vectorizer5.get_feature_names())
print("Shape of matrix after one hot encodig ",X_train_project_grade_category.shape)
feature_names_bow.extend(vectorizer5.get_feature_names())
feature_names_tfidf.append(vectorizer5.get_feature_names())
```

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
Shape of matrix after one hot encodig  (53531, 4)
```

In [35]:

```
X_train_prev_proj=X_train['teacher_number_of_previously_posted_projects'][:,np.newaxis]
X_test_prev_proj=X_test['teacher_number_of_previously_posted_projects'][:,np.newaxis]
X_cv_prev_proj=X_cv['teacher_number_of_previously_posted_projects'][:,np.newaxis]
feature_names_bow.append('teacher_number_of_previously_posted_projects')
feature_names_tfidf.append('teacher_number_of_previously_posted_projects')
```

Vectorize the Numerical Features - price

In [36]:

```
# Check before vectorizing
print("X_train['price'].shape-->",X_train['price'].shape)

print("X_train['price'].ndim-->",X_train['price'].ndim)

print("before normalize-->\n",X_train['price'][:10])
print("=="*25)
print("before normalize-->\n",X_test['price'][:10])
print("=="*25)
print("before normalize-->\n",X_cv['price'][:10])
```

```
X_train['price'].shape--> (53531,)
X_train['price'].ndim--> 1
before normalize-->
 266        105.18
106324    1863.96
89301       61.97
51090       64.46
32032      201.69
```

```
69153       65.98
85597      605.22
59877      317.47
419       1512.73
32534      777.94
Name: price, dtype: float64
==================================================
before normalize-->
 75155    237.41
77488    384.82
7803     329.88
56268    149.86
46902     28.99
66547     49.99
6472     109.80
50630    419.92
41045    338.98
25093    215.72
Name: price, dtype: float64
==================================================
before normalize-->
 58907     440.13
66108     299.95
55985     115.99
97535     437.34
6397      774.79
105728    499.00
29932     110.63
86353     197.42
51218     464.97
50960     109.89
Name: price, dtype: float64
```

In [37]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
# https://scikit-
learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html#sklearn.preprocessing.Norm
er

from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
from sklearn import preprocessing

X=X_train['price'].values.reshape(1,-1)
price_scalar = Normalizer().fit(X)

# Now standardize the data with above maen and variance.
X_train_price_standardized = price_scalar.transform(X_train['price'].values.reshape(1,-1))
X_test_price_standardized = price_scalar.transform(X_test['price'].values.reshape(1,-1))
X_cv_price_standardized = price_scalar.transform(X_cv['price'].values.reshape(1,-1))

# Make sure to reshape it back Else it will give error while merging
X_train_price_standardized=X_train_price_standardized.reshape(-1,1)

X_test_price_standardized=X_test_price_standardized.reshape(-1,1)

X_cv_price_standardized=X_cv_price_standardized.reshape(-1,1)


feature_names_bow.extend('price')
feature_names_tfidf.append('price')
print("X_train_price_standardized dimension is-->", X_train_price_standardized.ndim)
print("=="*25)
print("X_train_price_standardized shape is--> ", X_train_price_standardized.shape)
print("=="*25)
print("After normalize-->\n",X_test_price_standardized[:10])
print("=="*25)
print("After normalize-->\n",X_cv_price_standardized[:10])
print("=="*25)

feature_names_bow.extend('price')

feature_names_tfidf.append('price')
```

```
X_train_price_standardized dimension is--> 2
==================================================
X_train_price_standardized shape is-->  (53531, 1)
==================================================
After normalize-->
 [[ 0.00273872]
 [ 0.00443921]
 [ 0.00380544]
 [ 0.00172876]
 [ 0.00033442]
 [ 0.00057668]
 [ 0.00126663]
 [ 0.00484412]
 [ 0.00391041]
 [ 0.00248851]]
==================================================
After normalize-->
 [[ 0.00616436]
 [ 0.00420103]
 [ 0.00162453]
 [ 0.00612528]
 [ 0.01085153]
 [ 0.00698888]
 [ 0.00154946]
 [ 0.00276502]
 [ 0.00651226]
 [ 0.00153909]]
==================================================
```

Vectorize the Numerical Features - quantity

In [38]:

```
X=X_train['quantity'].values.reshape(1,-1)
quantity_scalar = Normalizer().fit(X)

# Now standardize the data with above maen and variance.
X_train_quantity_standardized = quantity_scalar.transform(X_train['quantity'].values.reshape(1,-1))
X_test_quantity_standardized = quantity_scalar.transform(X_test['quantity'].values.reshape(1,-1))
X_cv_quantity_standardized = quantity_scalar.transform(X_cv['quantity'].values.reshape(1,-1))

X_train_quantity_standardized =X_train_quantity_standardized.reshape(-1,1)
X_test_quantity_standardized = X_test_quantity_standardized.reshape(-1,1)
X_cv_quantity_standardized =X_cv_quantity_standardized.reshape(-1,1)

feature_names_bow.extend('quantity')
feature_names_tfidf.append('quantity')
```

Vectorize the Numerical Features - essay count

In [39]:

```
X=X_train['essay_count'].values.reshape(1,-1)
count_scalar = Normalizer().fit(X)


# Now standardize the data with above maen and variance.
X_train_essay_count_standardized = count_scalar.transform(X_train['essay_count'].values.reshape(1,
-1))
X_test_essay_count_standardized = count_scalar.transform(X_test['essay_count'].values.reshape(1,-1)
)
X_cv_essay_count_standardized = count_scalar.transform(X_cv['essay_count'].values.reshape(1,-1))

X_train_essay_count_standardized = X_train_essay_count_standardized.reshape(-1,1)
X_test_essay_count_standardized =X_test_essay_count_standardized.reshape(-1,1)
X_cv_essay_count_standardized = X_cv_essay_count_standardized.reshape(-1,1)

feature_names_bow.extend('essay_count')
feature_names_tfidf.append('essay_count')
```

Vectorize the Numerical Features - title count

In [40]:

```
X=X_train['title_count'].values.reshape(1,-1)
count_scalar = Normalizer().fit(X)

# Now standardize the data with above maen and variance.
X_train_title_count_standardized = count_scalar.transform(X_train['title_count'].values.reshape(1,
-1))
X_test_title_count_standardized = count_scalar.transform(X_test['title_count'].values.reshape(1,-1)
)
X_cv_title_count_standardized = count_scalar.transform(X_cv['title_count'].values.reshape(1,-1))

X_train_title_count_standardized = X_train_title_count_standardized.reshape(-1,1)
X_test_title_count_standardized = X_test_title_count_standardized.reshape(-1,1)
X_cv_title_count_standardized = X_cv_title_count_standardized.reshape(-1,1)

feature_names_bow.extend('title_count')
feature_names_tfidf.append('title_count')
```

Vectorizing Text data

Bag of word

In [41]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer6 = CountVectorizer(min_df=10,ngram_range=(1,2))
vectorizer6.fit(X_train['essay'])

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow=vectorizer6.transform(X_train['essay'].values)
X_test_essay_bow=vectorizer6.transform(X_test['essay'].values)
X_cv_essay_bow=vectorizer6.transform(X_cv['essay'].values)


print("Shape of matrix after one hot encodig ",X_train_essay_bow.shape)
feature_names_bow.extend(vectorizer6.get_feature_names())
```

```
Shape of matrix after one hot encodig  (53531, 104460)
```

In [42]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer7 = CountVectorizer(min_df=5)
vectorizer7.fit(X_train['Cleaned_title'])

# we use the fitted CountVectorizer to convert the text to vector
X_train_cleaned_title_bow=vectorizer7.transform(X_train['Cleaned_title'].values)
X_test_cleaned_title_bow=vectorizer7.transform(X_test['Cleaned_title'].values)
X_cv_cleaned_title_bow=vectorizer7.transform(X_cv['Cleaned_title'].values)


print("Shape of matrix after one hot encodig ",X_train_cleaned_title_bow.shape)
feature_names_bow.extend(vectorizer7.get_feature_names())
```

```
Shape of matrix after one hot encodig  (53531, 3334)
```

TFIDF vectorizer

In [43]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer8 = TfidfVectorizer(min_df=10,ngram_range=(1,2))
vectorizer8.fit(X_train['essay'])
```

```
# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_tfidf=vectorizer8.transform(X_train['essay'].values)
X_test_essay_tfidf=vectorizer8.transform(X_test['essay'].values)
X_cv_essay_tfidf=vectorizer8.transform(X_cv['essay'].values)


print("Shape of matrix after one hot encodig ",X_train_essay_tfidf.shape)
feature_names_tfidf.append(vectorizer8.get_feature_names())
```

```
Shape of matrix after one hot encodig  (53531, 104460)
```

In [44]:

```
X_train_prev_proj=X_train['teacher_number_of_previously_posted_projects'][:,np.newaxis]
X_test_prev_proj=X_test['teacher_number_of_previously_posted_projects'][:,np.newaxis]
X_cv_prev_proj=X_cv['teacher_number_of_previously_posted_projects'][:,np.newaxis]
feature_names_tfidf.append(vectorizer8.get_feature_names())
feature_names_tfidf.append(vectorizer8.get_feature_names())
```

In [45]:

```
# Similarly you can vectorize for title alsovectorizer = TfidfVectorizer(min_df=10)
vectorizer9 = TfidfVectorizer(min_df=5)

# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer9.fit(X_train['Cleaned_title'])

# we use the fitted CountVectorizer to convert the text to vector
X_train_cleaned_title_tfidf=vectorizer9.transform(X_train['Cleaned_title'].values)
X_test_cleaned_title_tfidf=vectorizer9.transform(X_test['Cleaned_title'].values)
X_cv_cleaned_title_tfidf=vectorizer9.transform(X_cv['Cleaned_title'].values)


print("Shape of matrix after one hot encodig ",X_train_cleaned_title_tfidf.shape)
feature_names_tfidf.append(vectorizer9.get_feature_names())
```

```
Shape of matrix after one hot encodig  (53531, 3334)
```

In [46]:

```
pd.DataFrame(X_train_cleaned_title_tfidf.toarray(),columns=vectorizer9.get_feature_names())
```

Out[46]:

	000	04	05	10	100	101	10th	11	12	123	...	york	young	youngest	youth	yummy	zen	zenergy	zone	zoo
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
10	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
11	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
13	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
14	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0

	000	04	05	10	100	101	10th	11	12	123	...	york	young	youngest	youth	yummy	zen	zenergy	zone	zoo
15	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
16	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
17	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
18	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
20	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
22	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
23	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
24	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
25	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
26	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
27	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
28	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
29	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
53501	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53502	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53503	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53504	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53505	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53506	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53507	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53508	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53509	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53510	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53511	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53512	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53513	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53514	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53515	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53516	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53517	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53518	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53519	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53520	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53521	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53522	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53523	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53524	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53525	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.531105	0.0	0.0	0.0	0.0	0.0
53526	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53527	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53528	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0
53529	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0

	000	04	05	10	100	101	10th	11	12	123	...	york	young	youngest	youth	yummy	zen	zenergy	zone	zoo
53530	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.0

53531 rows × 3334 columns

Merging all the above features

```
--we need to merge all the numerical vectors i.e catogorical, text, numerical vectors
```

In [47]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train_bow = hstack((X_train_clean_categories,
X_train_clean_sub_categories,X_train_skl_state,X_train_teacher_prefix,
            X_train_project_grade_category,X_train_prev_proj,X_train_price_standardized,X_train_qua
ntity_standardized,X_train_essay_count_standardized,X_train_title_count_standardized,
            X_train_essay_bow,X_train_cleaned_title_bow
            )).tocsr()


X_test_bow = hstack((X_test_clean_categories,
X_test_clean_sub_categories,X_test_skl_state,X_test_teacher_prefix,
            X_test_project_grade_category,X_test_prev_proj,X_test_price_standardized,X_test_quantit
y_standardized,X_test_essay_count_standardized,X_test_title_count_standardized,
            X_test_essay_bow,X_test_cleaned_title_bow
            )).tocsr()


X_cv_bow = hstack((X_cv_clean_categories,
X_cv_clean_sub_categories,X_cv_skl_state,X_cv_teacher_prefix,

X_cv_project_grade_category,X_cv_prev_proj,X_cv_price_standardized,X_cv_quantity_standardized,X_cv
_essay_count_standardized,X_cv_title_count_standardized,
            X_cv_essay_bow,X_cv_cleaned_title_bow
            )).tocsr()



print(X_train_bow.shape)
print(X_test_bow.shape)
print(X_cv_bow.shape)
```

```
(53531, 107898)
(32775, 107898)
(22942, 107898)
```

In [48]:

```
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X_train_tfidf = hstack((X_train_clean_categories, X_train_clean_sub_categories,X_train_skl_state,X
_train_teacher_prefix,
            X_train_project_grade_category,X_train_prev_proj,X_train_price_standardized,X_train_qua
ntity_standardized,X_train_essay_count_standardized,X_train_title_count_standardized,
            X_train_essay_tfidf,X_train_cleaned_title_tfidf
            )).tocsr()


X_test_tfidf = hstack((X_test_clean_categories,
X_test_clean_sub_categories,X_test_skl_state,X_test_teacher_prefix,
            X_test_project_grade_category,X_test_prev_proj,X_test_price_standardized,X_test_quantit
y_standardized,X_test_essay_count_standardized,X_test_title_count_standardized,
            X_test_essay_tfidf,X_test_cleaned_title_tfidf
            )).tocsr()


X_cv_tfidf = hstack((X_cv_clean_categories,
X_cv_clean_sub_categories,X_cv_skl_state,X_cv_teacher_prefix,
X_cv_project_grade_category,X_cv_prev_proj,X_cv_price_standardized,X_cv_quantity_standardized,X_cv
_essay_count_standardized,X_cv_title_count_standardized,
            X_cv_essay_tfidf,X_cv_cleaned_title_tfidf
            )).tocsr()



print(X_train_tfidf.shape)
print(X_test_tfidf.shape)
print(X_cv_tfidf.shape)
```

```
(53531, 107898)
(32775, 107898)
(22942, 107898)
```

Assignment 4: Naive Bayes

1. **Apply Multinomial NaiveBayes on these feature sets**

    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

2. **The hyper paramter tuning(find best Alpha)**

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

3. **Feature importance**

    - Find the top 10 features of positive class and top 10 features of negative class for both feature sets Set 1 and Set 2 using absolute values of `coef_` parameter of MultinomialNB and print their corresponding feature names

4. **Representation of results**

    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure. Here on X-axis you will have alpha values, since they have a wide range, just to represent those alpha values on the graph, apply log function on those alpha values.
    - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
    - Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

5. **Conclusion**

    - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link

Naive Bayes

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

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

Applying Naive Bayes on BOW, SET 1

In [49]:

```
# Please write all the code with proper documentation
from sklearn.metrics import roc_auc_score,auc
from sklearn.naive_bayes import MultinomialNB
summary=[]
roc_auc_score_cv_bow_dict={}
roc_auc_score_train_bow_dict={}

alpha=[0.00001,0.0001,0.001,0.01,0.1,1,10,100,1000,1000]
for i in tqdm(alpha):
    # create instance of model
    nb=MultinomialNB(alpha=i, class_prior=[0.5, 0.5])
     # fitting the model on crossvalidation train
    nb.fit(X_train_bow, y_train)

    # predict the response on the crossvalidation train
    pred_bow_cv = nb.predict_log_proba(X_cv_bow)

    #evaluate CV roc_auc
    roc_auc_cv =roc_auc_score(y_cv,pred_bow_cv[:,1])

    #insert into dict
    roc_auc_score_cv_bow_dict[i]=roc_auc_cv

     # fitting the model on crossvalidation train
    nb.fit(X_train_bow, y_train)

    # predict the response on the train
    pred_bow_train = nb.predict_log_proba(X_train_bow)

    #evaluate train roc_auc
    roc_auc_train =roc_auc_score(y_train,pred_bow_train[:,1])

    #insert into dict
    roc_auc_score_train_bow_dict[i]=roc_auc_train



print(roc_auc_score_cv_bow_dict)
print(roc_auc_score_train_bow_dict)
```

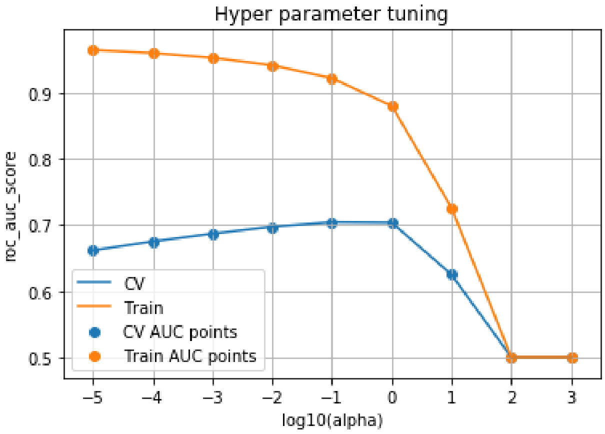
```
100%|████████████████████████████████████████████████████████████████████████████████| 10/10
[00:03<00:00,  2.67it/s]
```

```
{1e-05: 0.66152146104053844, 0.0001: 0.67491215209550282, 0.001: 0.68668909577056625, 0.01:
0.69720693535987377, 0.1: 0.70416558860397083, 1: 0.70350758071303043, 10: 0.62483456471668353, 10
0: 0.5, 1000: 0.5}
{1e-05: 0.96434304273262716, 0.0001: 0.95963603002349762, 0.001: 0.95252430968095447, 0.01:
0.94119848221110591, 0.1: 0.92163048383794655, 1: 0.87981516490576162, 10: 0.72581729991013488, 10
0: 0.5, 1000: 0.5}
```

Plot ROC_AUC_score VS different alpha values (Train and CV set

In [50]:

```
#https://stackoverflow.com/questions/37266341/plotting-a-python-dict-in-order-of-key-
values/37266356
import math

lists1 = sorted(roc_auc_score_cv_bow_dict.items())
x1, y1 = zip(*lists1) # unpack a list of pairs into two tuples

lists2 = sorted(roc_auc_score_train_bow_dict.items())
x2, y2 = zip(*lists2) # unpack a list of pairs into two tuples

x1=[math.log10(i) for i in list(x1)]
x2=[math.log10(i) for i in list(x2)]

plt.xlabel('log10(alpha)')
plt.ylabel('roc_auc_score')
plt.title('Hyper parameter tuning')

plt.plot(x1, y1,label="CV")
plt.plot(x2, y2,label='Train')
plt.scatter(x1, y1, label='CV AUC points')
plt.scatter(x2, y2, label='Train AUC points')

plt.grid()
plt.legend()
plt.show()
```



Find the best alpha value from the result

In [51]:

```
# This function returns the key (K) with max value in the dictionary
def find_highest_alpha(mydict):
    #print dictionary contents
    print("Dictionary Contents-->\n",mydict)
    print("="*50)
    #find maximum value from dixtionary
    maxval=max(mydict.values())

    # make a list of keys
    keys = list(mydict.keys())
    print("Keys in the list are-->\n", keys)
    print("="*50)
    # make a list of values
    vals = list(mydict.values())
    print("Values in the list are-->\n", vals)
    print("="*50)
    # find the index of max value and use it to find corresponding key
    k=(keys[vals.index(maxval)])
    print("Maximum k is {0} ".format(k))
    print("="*50)
    return(k)
```

Train the model on the optimal alpha value and run the Test Dataset

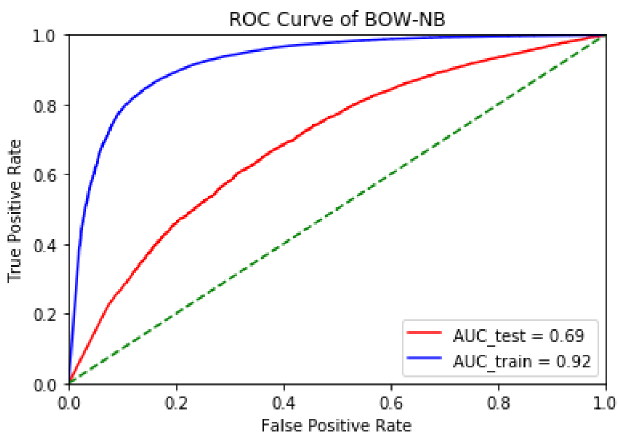
    -- Plot the ROC curve for BOW using the test and train Dataset

In [52]:

```
# train model on the best alpha
nb = MultinomialNB(alpha=find_highest_alpha(roc_auc_score_cv_bow_dict))

# fitting the model on crossvalidation train
nb.fit(X_train_bow, y_train)

# predict the response on the crossvalidation train
pred_bow_test = nb.predict(X_test_bow) # **we will use it in confusion matrix
pred_bow_train = nb.predict(X_train_bow) # **we will use it in confusion matrix

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_bow_test_scores=nb.predict_log_proba(X_test_bow)
pred_bow_train_scores=nb.predict_log_proba(X_train_bow)
```

```
fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_bow_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_bow_train_scores[:, 1])

#calculated AUC
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)

#save ALPHA & roc_auc_test score (AUC of fpr_test and tpr_test) for final summary
summary.append(['BOW',find_highest_alpha(roc_auc_score_cv_bow_dict),roc_auc_test])

plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of BOW-NB')
plt.show()
```

```
Dictionary Contents-->
 {1e-05: 0.66152146104053844, 0.0001: 0.67491215209550282, 0.001: 0.68668909577056625, 0.01:
0.69720693535987377, 0.1: 0.70416558860397083, 1: 0.70350758071303043, 10: 0.62483456471668353, 10
0: 0.5, 1000: 0.5}
==================================================
Keys in the list are-->
 [1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]
==================================================
Values in the list are-->
 [0.66152146104053844, 0.67491215209550282, 0.68668909577056625, 0.69720693535987377,
0.70416558860397083, 0.70350758071303043, 0.62483456471668353, 0.5, 0.5]
==================================================
Maximum k is 0.1
==================================================
Dictionary Contents-->
 {1e-05: 0.66152146104053844, 0.0001: 0.67491215209550282, 0.001: 0.68668909577056625, 0.01:
0.69720693535987377, 0.1: 0.70416558860397083, 1: 0.70350758071303043, 10: 0.62483456471668353, 10
0: 0.5, 1000: 0.5}
==================================================
Keys in the list are-->
 [1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]
==================================================
Values in the list are-->
 [0.66152146104053844, 0.67491215209550282, 0.68668909577056625, 0.69720693535987377,
0.70416558860397083, 0.70350758071303043, 0.62483456471668353, 0.5, 0.5]
==================================================
Maximum k is 0.1
==================================================
```



Get the confusion matrix for the BOW - NB|

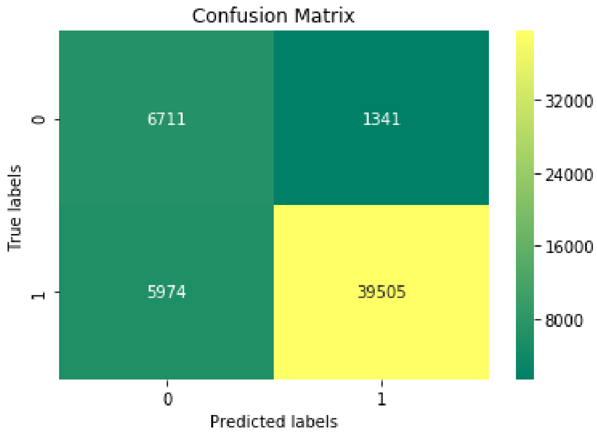
```
In [53]:
```

```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
# Make confusion matrix for y_train vs predicted(X_train_bow)
cm = confusion_matrix(y_train, pred_bow_train)
print(cm)

sns.heatmap(cm, annot=True, ax = ax,cmap="summer", fmt='g')
print("Training CM for BOW")
# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
```

```
[[ 6711  1341]
 [ 5974 39505]]
Training CM for BOW
```

Out[53]:

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



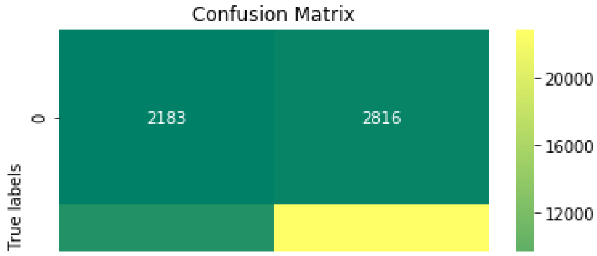
In [54]:

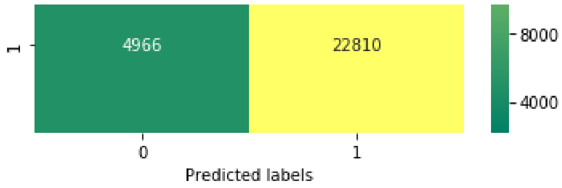
```
print("="*50)
print("Testing CM for BOW")
ax= plt.subplot()
# Make confusion matrix for y_test vs predicted(X_test_bow)
cm = confusion_matrix(y_test, pred_bow_test)
print(cm)

sns.heatmap(cm, annot=True, ax = ax,cmap="summer",fmt='g')
# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
```

```
==================================================
Testing CM for BOW
[[ 2183  2816]
 [ 4966 22810]]
```

Out[54]:

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





Top 10 important features of negative and positive class from SET 1

In [55]:

```
#for BOW
nb = MultinomialNB(alpha=0.1)# takes the k from the i th list value
nb.fit(X_train_bow, y_train)# fit the model

# now make a dictionary of all the probabilities fo the weights
bow_features_probs = []
for a in range(107893):
    bow_features_probs.append(nb.feature_log_prob_[0,a] )

print(len(bow_features_probs))

bow_features_names = []
for a in vectorizer1.get_feature_names() :# clean categories
    bow_features_names.append(a)
for a in vectorizer2.get_feature_names() :# sub categoreis
    bow_features_names.append(a)
for a in vectorizer3.get_feature_names() :#schooll state
    bow_features_names.append(a)
for a in vectorizer4.get_feature_names() :# teacher prefix
    bow_features_names.append(a)
for a in vectorizer5.get_feature_names() :# grade
    bow_features_names.append(a)
for a in vectorizer6.get_feature_names(): #titles bow
    bow_features_names.append(a)
for a in vectorizer7.get_feature_names(): # essays bow
    bow_features_names.append(a)
print( len(bow_features_names))
```

```
107893
107893
```

In [56]:

```
#top 10 negatives
final_bow_features = pd.DataFrame({'feature_prob_estimates' : bow_features_probs, 'feature_names':
bow_features_names})
a =final_bow_features.sort_values(by = ['feature_prob_estimates'], ascending = False)
#print(final_bow_features.head(6))
a.head(10)
```

Out[56]:

	feature_names	feature_prob_estimates
99	00	-3.492694
86593	students 11th	-3.508108
77700	school 12	-4.608102
50707	learning 4th	-4.927367
16294	classroom 15	-5.080501
62735	not academically	-5.266354
49531	learn 3d	-5.272821
41375	help able	-5.302831
60526	narrative	-5.474316
56460	many activities	-5.509263

In [57]:

```
#top 10 Positives
# now make a dictionary of all the probabilityies fo the weights
bow_features_probs_pos = []
for a in range(107893):
    bow_features_probs_pos.append(nb.feature_log_prob_[1,a] )# negative feature probabilities
#len(bow_features_probs)
final_bow_features = pd.DataFrame({'feature_prob_estimates_pos' :
bow_features_probs_pos,'feature_names' : bow_features_names})
a =final_bow_features.sort_values(by = ['feature_prob_estimates_pos'], ascending = False)
a.head(10)
```

Out[57]:

	feature_names	feature_prob_estimates_pos
99	00	-3.022283
86593	students 11th	-3.513425
77700	school 12	-4.657063
50707	learning 4th	-5.019148
16294	classroom 15	-5.044793
62735	not academically	-5.316639
49531	learn 3d	-5.355976
41375	help able	-5.389735
56460	many activities	-5.530494
60526	narrative	-5.546108

Applying Naive Bayes on TFIDF, SET 2

Plot ROC_AUC_score VS different Alpha values (Train and CV set)

In [58]:

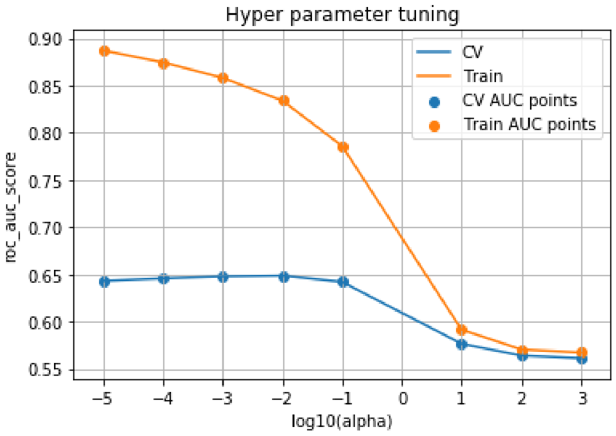
```
roc_auc_score_cv_tfidf_dict={}
roc_auc_score_train_tfidf_dict={}
alpha=[0.00001,0.0001,0.001,0.01,0.1,10,100,1000,1000]
for i in tqdm(alpha):
    # create instance of model
    nb=MultinomialNB(alpha=i, class_prior=[0.5, 0.5])

     # fitting the model on crossvalidation train
    nb.fit(X_train_tfidf, y_train)

    # predict the response on the crossvalidation train
    pred_tfidf_cv = nb.predict_log_proba(X_cv_tfidf)

    #evaluate CV roc_auc
    roc_auc_cv =roc_auc_score(y_cv,pred_tfidf_cv[:,1])

    #insert into dict
    roc_auc_score_cv_tfidf_dict[i]=roc_auc_cv

     # fitting the model on crossvalidation train
    nb.fit(X_train_tfidf, y_train)

    # predict the response on the train
    pred_tfidf_train = nb.predict_log_proba(X_train_tfidf)

    #evaluate train roc_auc
    roc_auc_train =roc_auc_score(y_train,pred_tfidf_train[:,1])

    #insert into dict
    roc_auc_score_train_tfidf_dict[i]=roc_auc_train



print(roc_auc_score_cv_tfidf_dict)
print(roc_auc_score_train_tfidf_dict)
```

```
100%|████████████████████████████████████████████████████████████████████████████████████| 9/9 [00
:04<00:00,  2.24it/s]
```

```
{1e-05: 0.64300188410186754, 0.0001: 0.64557257120445488, 0.001: 0.64777592787970784, 0.01:
0.6485837146308977, 0.1: 0.6419798004051076, 10: 0.57642272502802916, 100: 0.56409431445455027, 10
00: 0.56121781810733284}
{1e-05: 0.8870378050816311, 0.0001: 0.87454192677126585, 0.001: 0.85805539488607596, 0.01:
0.83401273284371924, 0.1: 0.78589948798803066, 10: 0.59132559224121028, 100: 0.57043413921998487,
1000: 0.56722502965535693}
```

In [59]:

```
#https://stackoverflow.com/questions/37266341/plotting-a-python-dict-in-order-of-key-
values/37266356
import math

lists1 = sorted(roc_auc_score_cv_tfidf_dict.items())
x1, y1 = zip(*lists1) # unpack a list of pairs into two tuples
lists2 = sorted(roc_auc_score_train_tfidf_dict.items())
x2, y2 = zip(*lists2) # unpack a list of pairs into two tuples

x1=[math.log10(i) for i in list(x1)]
x2=[math.log10(i) for i in list(x2)]

plt.xlabel('log10(alpha)')
plt.ylabel('roc_auc_score')
plt.title('Hyper parameter tuning')

plt.plot(x1, y1,label="CV")
plt.plot(x2, y2,label='Train')

plt.scatter(x1, y1, label='CV AUC points')
plt.scatter(x2, y2, label='Train AUC points')

plt.grid()
plt.legend()
plt.show()
```



Find the best alpha value from the result

In [60]:

```
print(find_highest_alpha(roc_auc_score_cv_tfidf_dict))
```

```
Dictionary Contents-->
 {1e-05: 0.64300188410186754, 0.0001: 0.64557257120445488, 0.001: 0.64777592787970784, 0.01:
0.6485837146308977, 0.1: 0.6419798004051076, 10: 0.57642272502802916, 100: 0.56409431445455027, 10
00: 0.56121781810733284}
```

```
00: 0.56121781810733284}
==================================================
Keys in the list are-->
 [1e-05, 0.0001, 0.001, 0.01, 0.1, 10, 100, 1000]
==================================================
Values in the list are-->
 [0.64300188410186754, 0.64557257120445488, 0.64777592787970784, 0.6485837146308977,
0.6419798004051076, 0.57642272502802916, 0.56409431445455027, 0.56121781810733284]
==================================================
Maximum k is 0.01
==================================================
0.01
```

Train the model on the optimal alpha value and run the Test Dataset

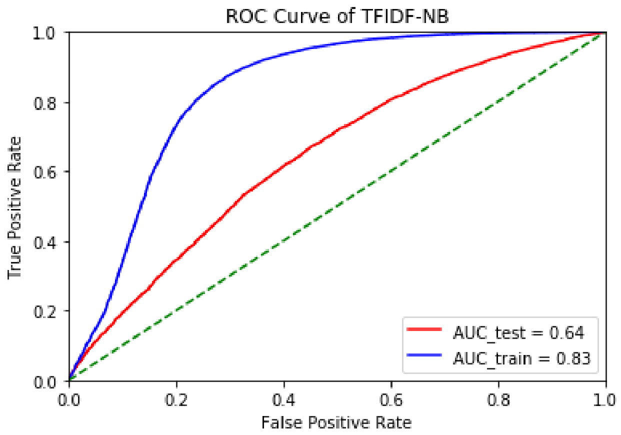
```
-- Plot the ROC curve for TFIDF using the test and train Dataset
```

In [61]:

```python
# train model on the best k
nb = MultinomialNB(alpha=find_highest_alpha(roc_auc_score_cv_tfidf_dict))

# fitting the model on crossvalidation train
nb.fit(X_train_tfidf, y_train)

# predict the response on the crossvalidation train
pred_tfidf_test = nb.predict(X_test_tfidf) # we will use this in confusion matrix

pred_tfidf_train = nb.predict(X_train_tfidf) # we will use this in confusion matrix

print("pred_tfidf_train-->", pred_tfidf_train)

print("pred_tfidf_test-->", pred_tfidf_test)

#https://stackoverflow.com/questions/52910061/implementing-roc-curves-for-k-nn-machine-learning-al
gorithm-using-python-and-scip_scores = knn.predict_proba(X_test)
pred_tfidf_test_scores=nb.predict_log_proba(X_test_tfidf)
pred_tfidf_train_scores=nb.predict_log_proba(X_train_tfidf)

fpr_test, tpr_test, threshold_test = roc_curve(y_test, pred_tfidf_test_scores[:, 1])
fpr_train, tpr_train, threshold_train = roc_curve(y_train, pred_tfidf_train_scores[:, 1])

#calculated AUC
roc_auc_test = auc(fpr_test, tpr_test)
roc_auc_train = auc(fpr_train, tpr_train)

#save ALPHA & roc_auc_test score (AUC of fpr_test and tpr_test) for final summary
summary.append(['TFIDF',find_highest_alpha(roc_auc_score_cv_tfidf_dict),roc_auc_test])

plt.title('Receiver Operating Characteristic')
plt.plot(fpr_test, tpr_test, 'r', label = 'AUC_test = %0.2f' % roc_auc_test)
plt.plot(fpr_train, tpr_train, 'b', label = 'AUC_train = %0.2f' % roc_auc_train)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'g--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title('ROC Curve of TFIDF-NB')
plt.show()
```

```
Dictionary Contents-->
 {1e-05: 0.64300188410186754, 0.0001: 0.64557257120445488, 0.001: 0.64777592787970784, 0.01:
0.6485837146308977, 0.1: 0.6419798004051076, 10: 0.57642272502802916, 100: 0.56409431445455027, 10
00: 0.56121781810733284}
==================================================
Keys in the list are-->
 [1e-05, 0.0001, 0.001, 0.01, 0.1, 10, 100, 1000]
==================================================
Values in the list are-->
 [0.64300188410186754, 0.64557257120445488, 0.64777592787970784, 0.6485837146308977,
0.6419798004051076, 0.57642272502802916, 0.56409431445455027, 0.56121781810733284]
==================================================
Maximum k is 0.01
```

```
==================================================
pred_tfidf_train--> [1 1 1 ..., 0 1 0]
pred_tfidf_test--> [1 1 1 ..., 1 0 1]
Dictionary Contents-->
 {1e-05: 0.6430018841018675 4, 0.0001: 0.64557257120445488, 0.001: 0.64777592787970784, 0.01:
0.6485837146308977, 0.1: 0.6419798004051076, 10: 0.57642272502802916, 100: 0.56409431445455027, 10
00: 0.56121781810733284}
==================================================
Keys in the list are-->
 [1e-05, 0.0001, 0.001, 0.01, 0.1, 10, 100, 1000]
==================================================
Values in the list are-->
 [0.64300188410186754, 0.64557257120445488, 0.64777592787970784, 0.6485837146308977,
0.6419798004051076, 0.57642272502802916, 0.56409431445455027, 0.56121781810733284]
==================================================
Maximum k is 0.01
==================================================
```



Get the confusion matrix for the TFIDF - NB

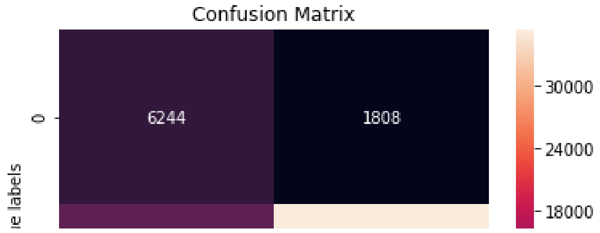
In [62]:

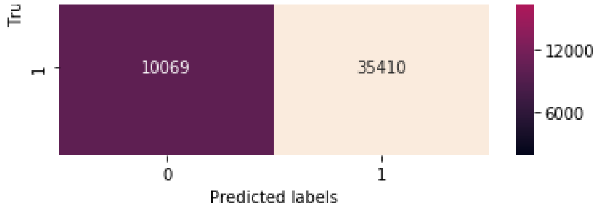
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()

# Make confusion matrix for y_train vs X_train_tfidf
cm = confusion_matrix(y_train, pred_tfidf_train)

print(cm)

sns.heatmap(cm, annot=True, ax = ax,fmt='g')
print("Training CM for TFID")
# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
```

```
[[ 6244  1808]
 [10069 35410]]
Training CM for TFID
```

Out[62]:

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





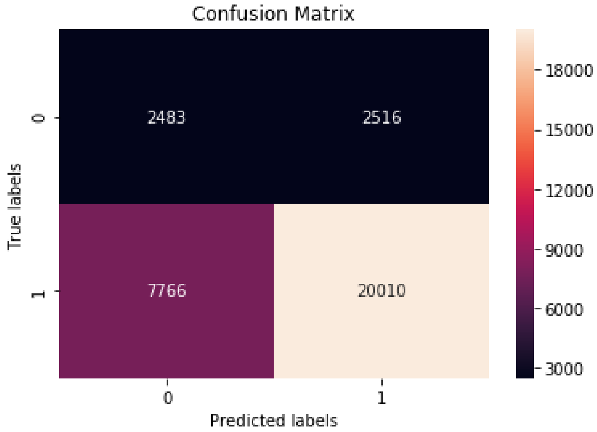
In [63]:

```
print("="*50)
print("Testing CM for TFIDF")
ax= plt.subplot()

# Make confusion matrix for y_test vs predicted(X_test_tfidf)
cm = confusion_matrix(y_test, pred_tfidf_test)
print(cm)
sns.heatmap(cm, annot=True, ax = ax,fmt='g')

# labels, title and ticks
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
```

```
==================================================
Testing CM for TFIDF
[[ 2483  2516]
 [ 7766 20010]]
```

Out[63]:

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



Top 10 important features of positive and negative class from SET 2

In [64]:

```
nb = MultinomialNB(alpha=.01)# takes the k from the i th list value
nb.fit(X_train_tfidf, y_train)# fit the model
# now make a dictionary of all the probabilityies fo the weights
tf_features_probs = []
for a in range(107893):# loop till (shape of data)
 tf_features_probs.append(nb.feature_log_prob_[0,a] )# negative feature probabilities
#len(bow_features_probs)
 tf_features_names = []
for a in vectorizer1.get_feature_names() :# clean categories
 tf_features_names.append(a)
for a in vectorizer2.get_feature_names() :# sub categoreis
 tf_features_names.append(a)
for a in vectorizer3.get_feature_names() :#schooll state
 tf_features_names.append(a)
for a in vectorizer4.get_feature_names() :# grade categoreis
 tf_features_names.append(a)
for a in vectorizer5.get_feature_names() :# teacher prefix
 tf_features_names.append(a)
 tf_features_names.append(a)
len(tf_features_names)

for a in vectorizer6.get_feature_names(): #titles tf_idf
 tf_features_names.append(a)
for a in vectorizer7.get_feature_names(): # essays tf_idf
 tf_features_names.append(a)


# top 10 negatives
final_tf_features = pd.DataFrame({'feature_prob_estimates' : tf_features_probs, 'feature_names' :t
f_features_names})
a =final_tf_features.sort_values(by = ['feature_prob_estimates'], ascending = False)
#print(final_bow_features.head(6))
a.head(10)
```

Out[64]:

	feature_names	feature_prob_estimates
99	00	-1.344570
92	Mrs	-3.970974
4	Literacy_Language	-4.143723
98	Grades_PreK_2	-4.195109
5	Math_Science	-4.208851
93	Ms	-4.285283
95	Grades_3_5	-4.407531
26	Literacy	-4.638182
28	Mathematics	-4.654252
27	Literature_Writing	-4.949653

In [65]:

```
#top 10 Positives
# now make a dictionary of all the probabilityies fo the weights
bow_features_probs_pos = []
for a in range(107893):
 bow_features_probs_pos.append(nb.feature_log_prob_[1,a] )# negative feature probabilities
#len(bow_features_probs)
final_bow_features = pd.DataFrame({'feature_prob_estimates_pos' :
bow_features_probs_pos,'feature_names' : bow_features_names})
a =final_bow_features.sort_values(by = ['feature_prob_estimates_pos'], ascending = False)
a.head(10)
```

Out[65]:

	feature_names	feature_prob_estimates_pos
99	00	-0.981388
92	Mrs	-4.102413
4	Literacy_Language	-4.177123
98	Grades_PreK_2	-4.361510
5	Math_Science	-4.436572
93	Ms	-4.490062
95	Grades_3_5	-4.530156
26	Literacy	-4.610011
28	Mathematics	-4.818778
27	Literature_Writing	-5.036141

Conclusions

In [66]:

```
# Please compare all your models using Prettytable library

from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Hyper parameter-ALPHA", "AUC"]

for each in summary:
    x.add_row(each)

print(x)
```

```
+------------+-----------------------+----------------+
| Vectorizer | Hyper parameter-ALPHA |      AUC       |
+------------+-----------------------+----------------+
|    BOW     |          0.1          | 0.694193047279 |
|   TFIDF    |          0.01         | 0.644586128487 |
+------------+-----------------------+----------------+
```