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 (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **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!</code`
`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.

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

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

```
C:\Users\Tarun Makkar\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning: paramiko missing, opening SSH/SCP/
SFTP paths will be disabled.  `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip install paramiko` to suppress')
C:\Users\Tarun Makkar\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunkiz
e to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [3]:
```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

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']
```

In [5]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[5]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [6]:
```
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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Sci
            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]))
```

1.3 preprocessing of `project_subject_subcategories`

In [7]:
```python
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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&", "Sci
            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]))
```

1.3 Text preprocessing

In [8]:
```python
# 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 [9]:
```python
project_data.head(2)
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	Educati Suppor Eng Learnei Hc
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	Wan Projecto Hui Leari

In [10]:
```python
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [11]:
```python
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. We are a melting pot o
f refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 
languages represented in our English Learner program with students at every level of mastery.  We also have over 40 cou
ntries represented with the families within our school.  Each student brings a wealth of knowledge and experiences to u
s that open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits of your worl
d.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home that begs for more resources.  Man
y times our parents are learning to read and speak English along side of their children.  Sometimes this creates barrie
rs 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 a
t 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 Learner Teacher and will be sen
t home regularly to watch.  The videos are to help the child develop early reading skills.\r\n\r\nParents that do not h
ave 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's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the tim
e. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority st
udents. \r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is a wh
ole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big festival with 
crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the har
d work put in during the school year, with a dunk tank being the most popular activity.My students will use these five 
brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in t
he classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During 
independent reading time they will be used as special chairs students will each use on occasion. I will utilize them in 
place of chairs at my small group tables during math and reading times. The rest of the day they will be used by the st
udents who need the highest amount of movement in their life in order to stay focused on school.\r\n\r\nWhenever asked 
what the classroom is missing, my students always say more Hokki Stools. They can't get their fill of the 5 stools we a
lready have. When the students are sitting in group with me on the Hokki Stools, they are always moving, but at the sam
e time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be take
n. There are always students who head over to the kidney table to get one of the stools who are disappointed as there a
re not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise 
that allow my students to do desk work and move at the same time. These stools will help students to meet their 60 minu
tes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my studen
ts, these chairs will take away the barrier that exists in schools for a child who can't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher 
in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room 
for my students look forward to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls 
of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and 
reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls 
separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing a
ll the information and experiences and keep on wanting more.With these resources such as the comfy red throw pillows an
d the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom 
setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success i
n each and every child's education. The nautical photo props will be used with each child as they step foot into our cl
assroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them develope
d, and then hung in our classroom ready for their first day of 4th grade.  This kind gesture will set the tone before e
ven the first day of school! The nautical thank you cards will be used throughout the year by the students as they crea
te thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, 
inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our 
classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank yo
u!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine 
motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. 
\r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the st
udents receive free or reduced price lunch.  Despite their disabilities and limitations, my students love coming to sch
ool and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to groove and 
move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so t
hey say.Wobble chairs are the answer and I love then because they develop their core, which enhances gross motor and in 
Turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to sit and do worksheets. They wa
nt to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color a
nd shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old dese
rves.nannan
==================================================

The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. -
William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-American, making up the largest segm
ent of the student body. A typical school in Dallas is made up of 23.2% African-American students. Most of the students 
are on free or reduced lunch. We aren't receiving doctors, lawyers, or engineers children from rich backgrounds or neig
hborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smart, effe
ctive, efficient, and disciplined students with good character.In our classroom we can utilize the Bluetooth for swift 
transitions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the vol
ume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as meaningful. But wit
h the bluetooth speaker my students will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart w
ill allow me to have more room for storage of things that are needed for the day and has an extra part to it I can use.
```

```
The table top chart has all of the letter, words and pictures for students to learn about different letters and it is m
ore accessible.nannan
==================================================
```

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[20000])
print(sent)
print("="*50)
```

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

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

In [15]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
sent = re.sub(r'[0-9]+',' ',sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitive delays gross fine m
otor delays to autism They are eager beavers and always strive to work their hardest working past their limitations The
materials we have are the ones I seek out for my students I teach in a Title I school where most of the students receiv
e free or reduced price lunch Despite their disabilities and limitations my students love coming to school and come eag
er to learn and explore Have you ever felt like you had ants in your pants and you needed to groove and move as you wer
e in a meeting This is how my kids feel all the time The want to be able to move as they learn or so they say Wobble ch
airs are the answer and I love then because they develop their core which enhances gross motor and in Turn fine motor s
kills They also want to learn through games my kids do not want to sit and do worksheets They want to learn to count by
jumping and playing Physical engagement is the key to our success The number toss and color and shape mats can make tha
t happen My students will forget they are doing work and just have the fun a   year old deserves nannan
```

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', 'each', 'few', 'more',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', '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
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    sent = re.sub(r'[0-9]+',' ',sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:52<00:00, 2099.06it/s]
```

In [18]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[18]:
```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gross fine motor delays a
utism they eager beavers always strive work hardest working past limitations the materials ones i seek students i teach
title i school students receive free reduced price lunch despite disabilities limitations students love coming school c
ome eager learn explore have ever felt like ants pants needed groove move meeting this kids feel time the want able mov
e learn say wobble chairs answer i love develop core enhances gross motor turn fine motor skills they also want learn g
ames kids not want sit worksheets they want learn count jumping playing physical engagement key success the number toss
color shape mats make happen my students forget work fun year old deserves nannan'
```

1.4 Preprocessing of `project_title`

In [19]:
```
# similarly you can preprocess the titles also
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^0-9A-Za-z]+', ' ', sent)
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords and len(e)>2)
    preprocessed_title.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:02<00:00, 42523.98it/s]
```

In [20]:
```
project_data['preprocessed_essays']=preprocessed_essays
project_data['preprocessed_title']=preprocessed_title
```

In [21]:
```
project_data.drop(['project_title','project_essay_1','project_essay_2','project_essay_3','project_essay_4','essay'],inpl
```

In [22]:
```
project_data.head()
```

Out[22]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades PreK-2	opp
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grades 6-8	My stude
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grades 6-8	My gua
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades PreK-2	My stud in
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades PreK-2	My stu pr

1.5 Preparing data for models

In [23]:
```
project_data.columns
```

Out[23]:
```
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', 'preprocessed_essays',
       'preprocessed_title'],
      dtype='object')
```

Preprocessing numerical feature i.e price

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')
```

Checking for missing values

In [25]:
```
project_data.isnull().sum()
```

Out[25]:
```
Unnamed: 0                                      0
id                                              0
teacher_id                                      0
teacher_prefix                                  3
school_state                                    0
project_submitted_datetime                      0
project_grade_category                          0
project_resource_summary                        0
teacher_number_of_previously_posted_projects    0
project_is_approved                             0
clean_categories                                0
clean_subcategories                             0
preprocessed_essays                             0
preprocessed_title                              0
price                                           0
quantity                                        0
dtype: int64
```

In [26]:
```
project_data['teacher_prefix'].describe()
```

Out[26]:
```
count     109245
unique         5
top         Mrs.
freq       57269
Name: teacher_prefix, dtype: object
```

In [27]:
```
project_data['teacher_prefix'].replace(np.nan,'Mrs.',inplace=True)
```

In [28]:
```
project_data.isnull().sum()
```

Out[28]:
```
Unnamed: 0                                      0
id                                              0
teacher_id                                      0
teacher_prefix                                  0
school_state                                    0
project_submitted_datetime                      0
project_grade_category                          0
project_resource_summary                        0
teacher_number_of_previously_posted_projects    0
project_is_approved                             0
clean_categories                                0
clean_subcategories                             0
preprocessed_essays                             0
preprocessed_title                              0
price                                           0
quantity                                        0
dtype: int64
```

Splitting before vectorizing

In [29]:
```
#importing required libaries
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
```

In [30]:
```
y=project_data['project_is_approved']
X=project_data.drop('project_is_approved',axis=1)
X_tr, X_test, y_tr, y_test = train_test_split(X,y, test_size=0.2, random_state=0)
```

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
```

1.5.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/ (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/)

In [31]:
```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_category = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
categoriesxtr_one_hot = vectorizer_category.fit_transform(X_tr['clean_categories'].values)
print(vectorizer_category.get_feature_names())
print("Shape of matrix after one hot encodig ",categoriesxtr_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Sci
ence', 'Literacy_Language']
Shape of matrix after one hot encodig  (87398, 9)
```

In [32]:
```
categoriesxtest_one_hot = vectorizer_category.transform(X_test['clean_categories'].values)

print("Shape of matrix after one hot encodig ",categoriesxtest_one_hot.shape)
```

```
Shape of matrix after one hot encodig  (21850, 9)
```

In [33]:
```
# we use count vectorizer to convert the values into one
vectorizer_sub_category = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
sub_categoriesxtr_one_hot = vectorizer_sub_category.fit_transform(X_tr['clean_subcategories'].values)
print(vectorizer_sub_category.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categoriesxtr_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'Fo
reignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducatio
n', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopmen
t', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds',
'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (87398, 30)
```

In [34]:
```
sub_categoriesxtest_one_hot = vectorizer_sub_category.transform(X_test['clean_subcategories'].values)
print(vectorizer_sub_category.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categoriesxtest_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'Fo
reignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducatio
n', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopmen
t', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds',
'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (21850, 30)
```

In [35]:
```
vectorizer_teacher_prefix = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
teacher_prefixxtr_one_hot = vectorizer_teacher_prefix.fit_transform(X_tr['teacher_prefix'].values)

print("Shape of matrix after one hot encodig ",teacher_prefixxtr_one_hot.shape)
```

```
Shape of matrix after one hot encodig  (87398, 30)
```

In [36]:
```
teacher_prefixxtest_one_hot = vectorizer_teacher_prefix.transform(X_test['teacher_prefix'].values)

print("Shape of matrix after one hot encodig ",teacher_prefixxtest_one_hot.shape)
```

```
Shape of matrix after one hot encodig  (21850, 30)
```

In [37]:
```
project_grade_category_list=list(X_tr['project_grade_category'].unique())
vectorizer_grade = CountVectorizer(vocabulary=project_grade_category_list, lowercase=False, binary=True)
vectorizer_grade.fit(X_tr['project_grade_category'].values)

project_grade_categoryxtr_one_hot = vectorizer_grade.transform(X_tr['project_grade_category'].values)
print("Shape of matrix after one hot encodig ",project_grade_categoryxtr_one_hot.shape)
```

```
Shape of matrix after one hot encodig  (87398, 4)
```

In [38]:
```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
project_grade_categoryxtest_one_hot = vectorizer_grade.transform(X_test['project_grade_category'].values)
print("Shape of matrix after one hot encodig ",project_grade_categoryxtest_one_hot.shape)
```

```
Shape of matrix after one hot encodig  (21850, 4)
```

In [39]:
```
school_state_list=list(project_data['school_state'].unique())
vectorizer_school_state = CountVectorizer(vocabulary=school_state_list, lowercase=False, binary=True)
vectorizer_school_state.fit(X_tr['school_state'].values)
print(vectorizer_school_state.get_feature_names())


school_statextr_one_hot = vectorizer_school_state.transform(X_tr['school_state'].values)
print("Shape of matrix after one hot encodig ",school_statextr_one_hot.shape)
```

```
['IN', 'FL', 'AZ', 'KY', 'TX', 'CT', 'GA', 'SC', 'NC', 'CA', 'NY', 'OK', 'MA', 'NV', 'OH', 'PA', 'AL', 'LA', 'VA', 'A
R', 'WA', 'WV', 'ID', 'TN', 'MS', 'CO', 'UT', 'IL', 'MI', 'HI', 'IA', 'RI', 'NJ', 'MO', 'DE', 'MN', 'ME', 'WY', 'ND',
'OR', 'AK', 'MD', 'WI', 'SD', 'NE', 'NM', 'DC', 'KS', 'MT', 'NH', 'VT']
Shape of matrix after one hot encodig  (87398, 51)
```

In [40]:
```
school_statextest_one_hot = vectorizer_school_state.transform(X_test['school_state'].values)
print("Shape of matrix after one hot encodig ",school_statextest_one_hot.shape)
```

```
Shape of matrix after one hot encodig  (21850, 51)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [41]:
```python
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_essay = CountVectorizer(min_df=10)
vectorizer_essay.fit(X_tr['preprocessed_essays'])
print(vectorizer_essay.get_feature_names())
processed_essayxtr_bow = vectorizer_essay.transform(X_tr['preprocessed_essays'])
```

```
['aa', 'aaa', 'aac', 'ab', 'aba', 'abacus', 'abandon', 'abandoned', 'abc', 'abcmouse', 'abcs', 'abcya', 'abdominal',
'abilities', 'ability', 'abject', 'abl', 'able', 'abled', 'abound', 'abounds', 'about', 'above', 'abraham', 'abreas
t', 'abroad', 'abs', 'absence', 'absences', 'absent', 'absentee', 'absenteeism', 'absolute', 'absolutely', 'absorb',
'absorbed', 'absorbing', 'absorbs', 'abstract', 'abstraction', 'abstractly', 'abundance', 'abundant', 'abundantly',
'abuse', 'abused', 'abusive', 'abuzz', 'ac', 'academia', 'academic', 'academically', 'academics', 'academies', 'acad
emy', 'accelerate', 'accelerated', 'accelerating', 'acceleration', 'accent', 'accents', 'accentuate', 'accept', 'acc
eptable', 'acceptance', 'accepted', 'accepting', 'accepts', 'access', 'accessed', 'accesses', 'accessibility', 'acce
ssible', 'accessing', 'accessories', 'accessory', 'accident', 'accidental', 'accidentally', 'accidents', 'acclaime
d', 'acclimate', 'acclimated', 'acclimating', 'accolades', 'accommodate', 'accommodated', 'accommodates', 'accommoda
ting', 'accommodation', 'accommodations', 'accomodate', 'accompanied', 'accompanies', 'accompaniment', 'accompanimen
ts', 'accompany', 'accompanying', 'accomplish', 'accomplished', 'accomplishes', 'accomplishing', 'accomplishment',
'accomplishments', 'accordance', 'according', 'accordingly', 'account', 'accountability', 'accountable', 'accountin
g', 'accounts', 'accreditation', 'accredited', 'accumulate', 'accumulated', 'accumulating', 'accuracy', 'accurate',
'accurately', 'accustom', 'accustomed', 'ace', 'acer', 'ache', 'acheive', 'aches', 'achievable', 'achieve', 'achieve
d', 'achievement', 'achievements', 'achiever', 'achievers', 'achieves', 'achieving', 'aching', 'acid', 'acids', 'ack
nowledge', 'acknowledged', 'acknowledgement', 'acknowledges', 'acknowledging', 'acoustic', 'acoustics', 'acquainte
d', 'acquire', 'acquired', 'acquiring', 'acquisition', 'acre', 'acres', 'acronym', 'across', 'acrylic', 'acrylics',
'act', 'acted', 'acting', 'action', 'actions', 'activate', 'activated', 'activates', 'activating', 'activboard', 'ac
tive', 'actively', 'actives', 'activies', 'activism', 'activist', 'activists', 'activites', 'activities', 'activit
```

In [42]:
```python
print(vectorizer_essay.get_feature_names()[1754])
```

```
but
```

In [43]:
```python
processed_essayxtest_bow = vectorizer_essay.transform(X_test['preprocessed_essays'])

print("Shape of matrix after one hot encodig ",processed_essayxtest_bow.shape)
```

```
Shape of matrix after one hot encodig  (21850, 14996)
```

In [44]:
```python
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_title = CountVectorizer(min_df=10)
processed_titlextr_bow = vectorizer_title.fit_transform(X_tr['preprocessed_title'])
print("Shape of matrix after one hot encodig ",processed_titlextr_bow.shape)
```

```
Shape of matrix after one hot encodig  (87398, 2851)
```

In [45]:
```python
processed_titlextest_bow = vectorizer_title.transform(X_test['preprocessed_title'])
print("Shape of matrix after one hot encodig ",processed_titlextest_bow.shape)
```

```
Shape of matrix after one hot encodig  (21850, 2851)
```

1.5.2.2 TFIDF vectorizer

In [46]:
```python
vectorizer_tfidfessay = TfidfVectorizer(min_df=10)
vectorizer_tfidfessay.fit(X_tr['preprocessed_essays'])
processed_essayxtr_tfidf =vectorizer_tfidfessay.transform(X_tr['preprocessed_essays'])

print("Shape of matrix after one hot encodig ",processed_essayxtr_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (87398, 14996)
```

In [47]:
```python
processed_essayxtest_tfidf = vectorizer_tfidfessay.transform(X_test['preprocessed_essays'])

print("Shape of matrix after one hot encodig ",processed_essayxtest_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (21850, 14996)
```

In [48]:
```python
vectorizer_tfidftitle = TfidfVectorizer(min_df=10)
vectorizer_tfidftitle.fit(X_tr['preprocessed_title'])
processed_titlextr_tfidf = vectorizer_tfidftitle.transform(X_tr['preprocessed_title'])

print("Shape of matrix after one hot encodig ",processed_titlextr_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (87398, 2851)
```

In [49]:
```python
processed_titlextest_tfidf = vectorizer_tfidftitle.transform(X_test['preprocessed_title'])

print("Shape of matrix after one hot encodig ",processed_titlextest_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (21850, 2851)
```

1.5.3 Vectorizing Numerical features

In [50]:
```
# 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
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ... 399.   287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(X_tr['price'].values.reshape(-1,1)) # finding the mean and standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardizedxtr = price_scalar.transform(X_tr['price'].values.reshape(-1, 1))
```

```
Mean : 297.89565413396184, Standard deviation : 369.06453578992995
```

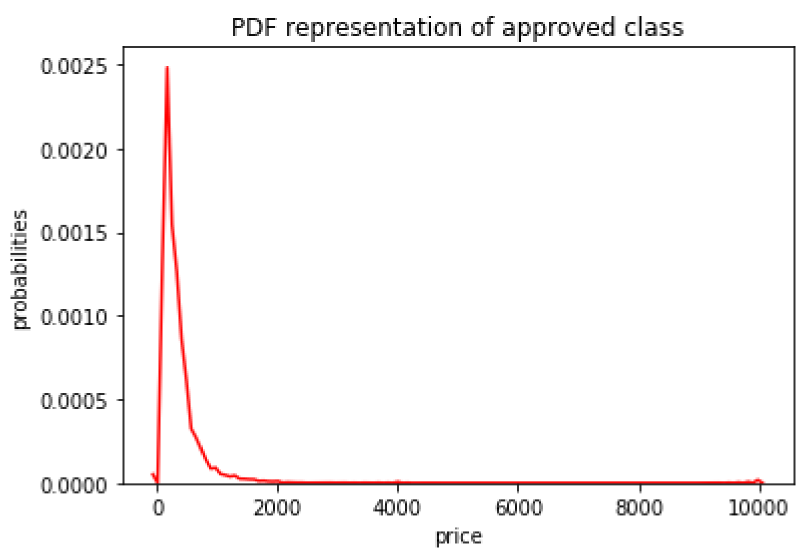
In [51]:
```
price_standardizedxtest = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
```

In [52]:
```
d1=project_data['price'][project_data['project_is_approved']==1]
```

In [53]:
```
d2=project_data['price'][project_data['project_is_approved']==0]
```

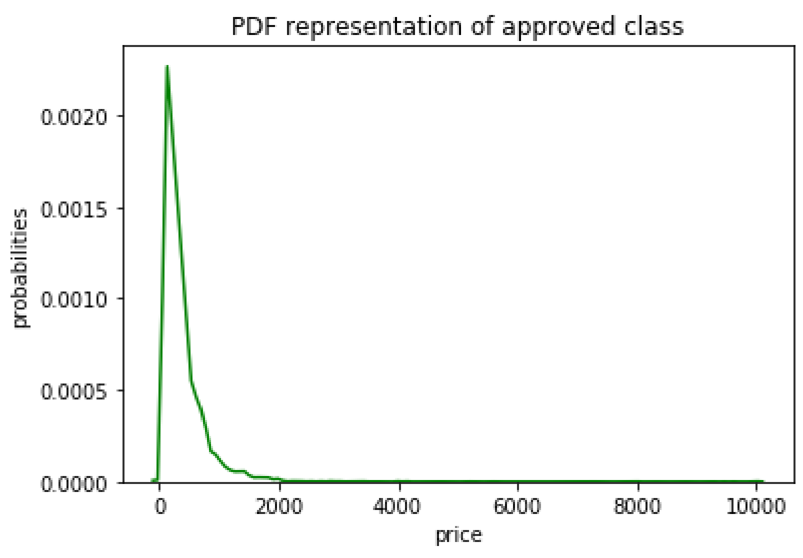
In [54]:
```
sns.distplot(d1,hist=False,color='r')
plt.ylabel('probabilities')
plt.title('PDF representation of approved class')
```

Out[54]:
```
Text(0.5, 1.0, 'PDF representation of approved class')
```



In [55]:
```
sns.distplot(d2,hist=False,color='g')
plt.ylabel('probabilities')
plt.title('PDF representation of approved class')
```

Out[55]:
```
Text(0.5, 1.0, 'PDF representation of approved class')
```



OBSERVATION:

- Both classes are not gaussian.So we have to assume that they are gaussian

1.5.4 Merging all the above features

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

In [56]:
```
# categorical features
print(categoriesxtr_one_hot.shape)
print(sub_categoriesxtr_one_hot.shape)
print(teacher_prefixxtr_one_hot.shape)
print(project_grade_categoryxtr_one_hot.shape)
print(school_statextr_one_hot.shape)

# text Features
print(processed_essayxtr_bow.shape)
print(processed_titlextr_bow.shape)

# numerical features
print(price_standardizedxtr.shape)
```

```
(87398, 9)
(87398, 30)
(87398, 30)
(87398, 4)
(87398, 51)
(87398, 14996)
(87398, 2851)
(87398, 1)
```

In [57]:
```
# 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 =hstack((categoriesxtr_one_hot, sub_categoriesxtr_one_hot, teacher_prefixxtr_one_hot,project_grade_categoryxtr_o
X_train.shape
```

Out[57]:
```
(87398, 17971)
```

In [58]:
```
X_train1 = hstack((categoriesxtr_one_hot, sub_categoriesxtr_one_hot, teacher_prefixxtr_one_hot,project_grade_categoryxtr
X_train1.shape
```

Out[58]:
```
(87398, 17971)
```

In [59]:
```
X_test = hstack((categoriesxtest_one_hot, sub_categoriesxtest_one_hot, teacher_prefixxtest_one_hot,project_grade_categor
X_test.shape
```

Out[59]:
```
(21850, 17971)
```

In [60]:
```
X_test1 = hstack((categoriesxtest_one_hot, sub_categoriesxtest_one_hot, teacher_prefixxtest_one_hot,project_grade_catego
X_test1.shape
```

Out[60]:
```
(21850, 17971)
```

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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) 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 (https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) 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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps. (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

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 (http://zetcode.com/python/prettytable/)

2. Naive Bayes

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

2.4.1 Applying Naive Bayes on BOW, SET 1

In [61]:
```
# for generating random aplha's between 10**-4 to 10**4
import random
a=[]
e=4
f=1
for g in range(5):
    for i in range(10):
        a.append(random.uniform(10**-e,0))
    e-=1

for g in range(5):
    for i in range(10):
        a.append(random.uniform(0,10**f))
    f+=1
```

In [62]:
```
from sklearn.naive_bayes import MultinomialNB
```

In [64]:
```
from sklearn.model_selection import GridSearchCV

parameters = {'alpha' : a }

NB = MultinomialNB(fit_prior=True,class_prior=[0.5,0.5])

estimator = GridSearchCV(NB,parameters,scoring='roc_auc',cv=4,verbose=22,n_jobs=4)
estimator.fit(X_train,y_tr)
```

```
Fitting 4 folds for each of 100 candidates, totalling 400 fits
```

In [65]:
```
estimator.best_estimator_
b = estimator.best_params_['alpha']
print(b)
```

```
0.31893343665050455
```

OBSERVATION :

- Optimal alpha is 0.31893343665050455

Top 10 features of set 1

In [69]:
```
neg_class_prob_sorted = mn1.feature_log_prob_[0, :].argsort()[::-1][0:10]
pos_class_prob_sorted = mn1.feature_log_prob_[1, :].argsort()[::-1][0:10]
print(neg_class_prob_sorted)
print(pos_class_prob_sorted)
```

```
[12989 11783  7721  8795  2423  9033  7717 13559  6332 13519]
[12989 11783  8795  7721  2423 13519 13559  9033  7717  6332]
```

In [70]:
```
m1=[vectorizer_essay.get_feature_names()[x-124] for x in neg_class_prob_sorted]
print(m1)

m2=[vectorizer_essay.get_feature_names()[x-124] for x in pos_class_prob_sorted]
print(m2)
```

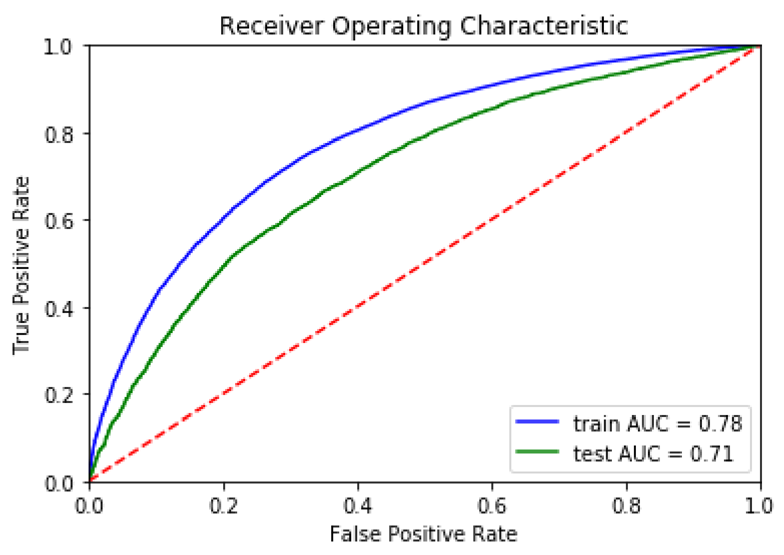
```
['students', 'school', 'learning', 'my', 'classroom', 'not', 'learn', 'they', 'help', 'the']
['students', 'school', 'my', 'learning', 'classroom', 'the', 'they', 'not', 'learn', 'help']
```

ROC and AUC

In [68]:
```
mn1=MultinomialNB(alpha=b,fit_prior=True,class_prior=[0.5,0.5])
mn1.fit(X_train, y_tr)
pred = mn1.predict_proba(X_train)
pred1=pred[:,1]
fpr1, tpr1, threshold1 = metrics.roc_curve(y_tr, pred1)
roc_auc1 = metrics.auc(fpr1, tpr1)
```

In [71]:
```
pred = mn1.predict_proba(X_test)
pred2=pred[:,1]
fpr2, tpr2, threshold2 = metrics.roc_curve(y_test, pred2)
roc_auc2 = metrics.auc(fpr2, tpr2)
```

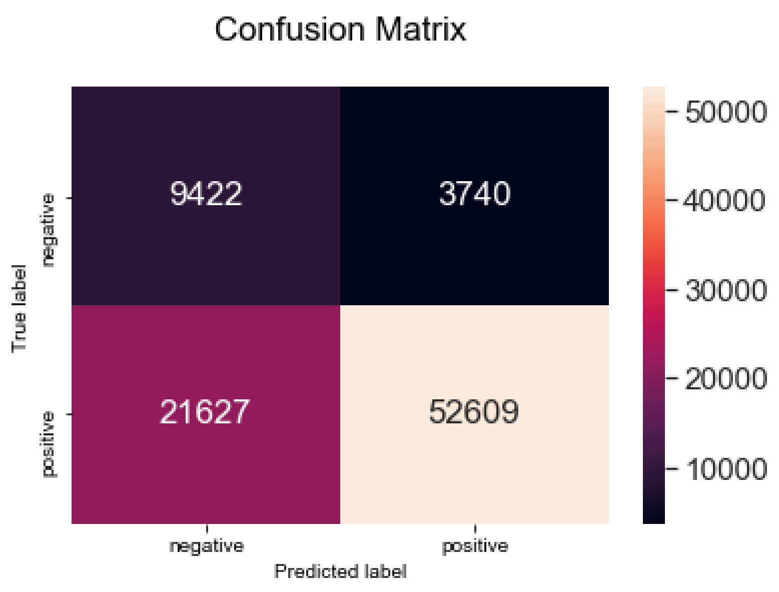
In [72]:
```
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr1, tpr1, 'b', label = 'train AUC = %0.2f' % roc_auc1)
plt.plot(fpr2, tpr2, 'g', label = 'test AUC = %0.2f' % roc_auc2)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [73]:
```
from sklearn.metrics import confusion_matrix
pred = mn1.predict(X_train)
pred_test = mn1.predict(X_test)
cm=metrics.confusion_matrix(y_tr,pred)
```

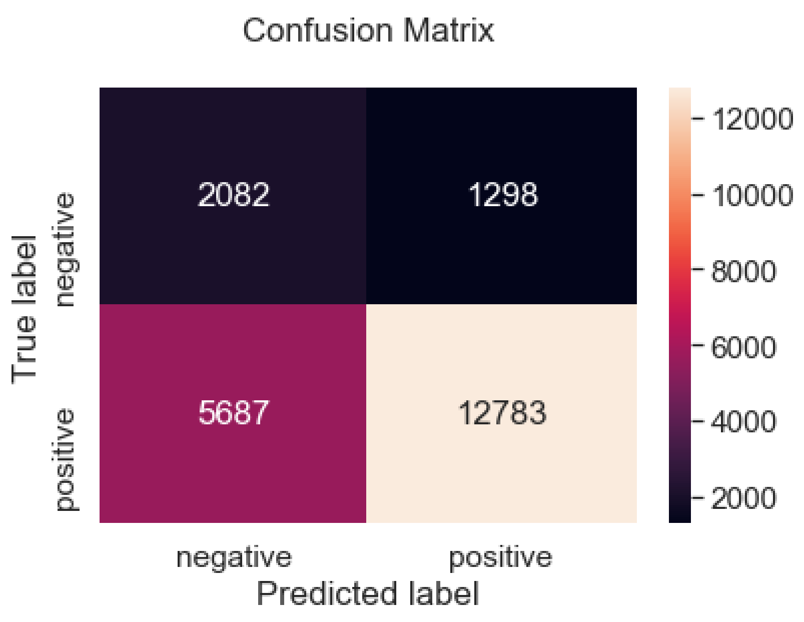
In [74]:
```
ax= plt.subplot()

sns.set(font_scale=1.4)
sns.heatmap(cm,annot=True,ax=ax,fmt='d')

ax.set_xlabel('Predicted label')
ax.set_ylabel('True label')
ax.xaxis.set_ticklabels(['negative','positive'])
ax.yaxis.set_ticklabels(['negative','positive'])
plt.title('Confusion Matrix\n')
```

Out[74]:
```
Text(0.5, 1.0, 'Confusion Matrix\n')
```



In [75]:
```
cm=metrics.confusion_matrix(y_test,pred_test)

ax= plt.subplot()

sns.set(font_scale=1.4)
sns.heatmap(cm,annot=True,ax=ax,fmt='d')

ax.set_xlabel('Predicted label')
ax.set_ylabel('True label')
ax.xaxis.set_ticklabels(['negative','positive'])
ax.yaxis.set_ticklabels(['negative','positive'])
plt.title('Confusion Matrix\n')
```

Out[75]:
```
Text(0.5, 1.0, 'Confusion Matrix\n')
```



2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [76]:
```
# for generating random aplha's between 10**-4 to 10**4
import random
a=[]
e=4
f=1
for g in range(5):
    for i in range(10):
        a.append(random.uniform(10**-e,0))
    e-=1

for g in range(5):
    for i in range(10):
        a.append(random.uniform(0,10**f))
    f+=1
```

In [78]:
```
from sklearn.model_selection import RandomizedSearchCV
parameters = {'alpha' : a }

NB = MultinomialNB(fit_prior=True,class_prior=[0.5,0.5])

estimator = RandomizedSearchCV(NB,parameters,scoring='roc_auc',cv=4,verbose=22,n_jobs=4)
estimator.fit(X_train,y_tr)
```

```
[Parallel(n_jobs=4)]: Done  27 tasks      | elapsed:    3.8s
[Parallel(n_jobs=4)]: Done  28 tasks      | elapsed:    3.8s
[Parallel(n_jobs=4)]: Done  29 tasks      | elapsed:    4.2s
[Parallel(n_jobs=4)]: Done  30 tasks      | elapsed:    4.3s
[Parallel(n_jobs=4)]: Done  31 tasks      | elapsed:    4.3s
[Parallel(n_jobs=4)]: Done  32 tasks      | elapsed:    4.4s
[Parallel(n_jobs=4)]: Done  33 tasks      | elapsed:    4.8s
[Parallel(n_jobs=4)]: Done  35 out of  40 | elapsed:    4.9s remaining:    0.6s
[Parallel(n_jobs=4)]: Done  37 out of  40 | elapsed:    5.2s remaining:    0.3s
[Parallel(n_jobs=4)]: Done  40 out of  40 | elapsed:    5.4s finished
```

Out[78]:
```
RandomizedSearchCV(cv=4, error_score='raise-deprecating',
          estimator=MultinomialNB(alpha=1.0, class_prior=[0.5, 0.5], fit_prior=True),
          fit_params=None, iid='warn', n_iter=10, n_jobs=4,
          param_distributions={'alpha': [8.803614010163736e-05, 7.34182781223851e-05, 7.823480907494368e-05, 5.14941
6141066446e-05, 6.0061426901662986e-05, 9.786673027581522e-05, 2.7459954303190787e-05, 2.773840254511295e-05, 4.1729
62860457247e-05, 1.4574304343832592e-05, 0.00022825029899185107, 0.00070379986...417, 428.1034570083109, 85311.34383
014531, 85190.93937019669, 1769.665848913049, 91006.71602810509]},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score='warn', scoring='roc_auc', verbose=22)
```

In [79]:
```
estimator.best_estimator_
b = estimator.best_params_['alpha']
print(b)
```

```
0.07007623159576255
```

OBSERVATION:

- Optimal alpha is 0.07007623159576255

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

In [81]:
```
# Please write all the code with proper documentation
neg_class_prob_sorted = mn1.feature_log_prob_[0, :].argsort()[::-1][:10]
pos_class_prob_sorted = mn1.feature_log_prob_[1, :].argsort()[::-1][:10]
```

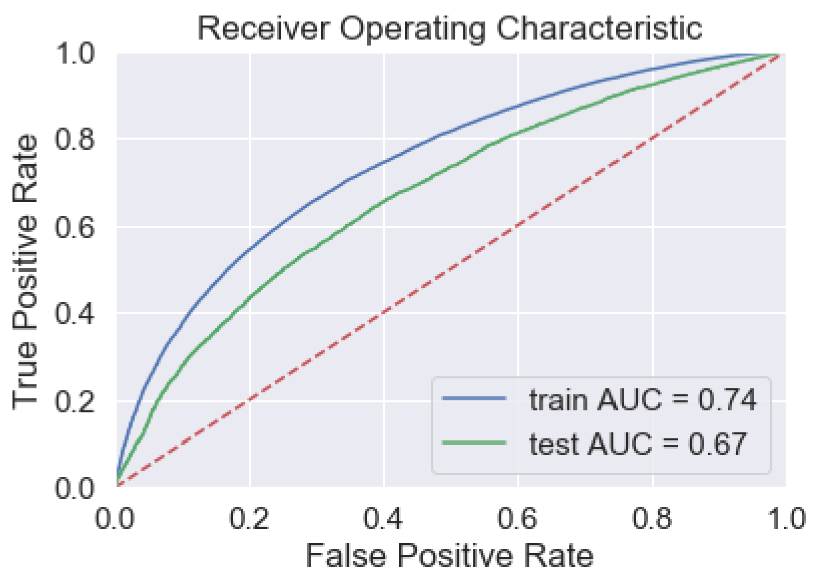
In [82]:
```
n1=[vectorizer_tfidfessay.get_feature_names()[x] for x in neg_class_prob_sorted]
print(n1)

n2=[vectorizer_tfidfessay.get_feature_names()[x] for x in pos_class_prob_sorted]
print(n2)
```

```
['abc', 'abandoned', 'abstract', 'absorbs', 'absorbing', 'abacus', 'absorbed', 'abandon', 'acclimated', 'summers']
['abc', 'abandoned', 'abstract', 'absorbs', 'absorbing', 'acclimated', 'summers', 'abandon', 'absorbed', 'abacus']
```

In [80]:
```
mn1=MultinomialNB(alpha=0.4965852145553745,fit_prior=True,class_prior=[0.5,0.5])
mn1.fit(X_train1, y_tr)
pred = mn1.predict_proba(X_train1)
pred1=pred[:,1]
fpr1, tpr1, threshold1 = metrics.roc_curve(y_tr, pred1)
roc_auc1 = metrics.auc(fpr1, tpr1)
```

In [83]:
```
pred = mn1.predict_proba(X_test1)
pred2=pred[:,1]
fpr2, tpr2, threshold2 = metrics.roc_curve(y_test, pred2)
roc_auc2 = metrics.auc(fpr2, tpr2)
```

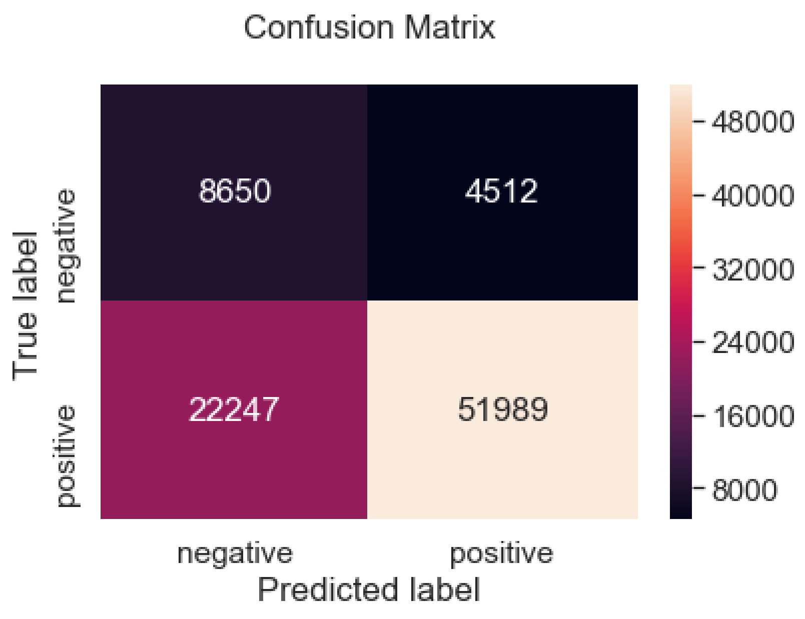
In [84]:
```
import matplotlib.pyplot as plt
plt.title('Receiver Operating Characteristic')
plt.plot(fpr1, tpr1, 'b', label = 'train AUC = %0.2f' % roc_auc1)
plt.plot(fpr2, tpr2, 'g', label = 'test AUC = %0.2f' % roc_auc2)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```



In [85]:
```
from sklearn.metrics import confusion_matrix
pred = mn1.predict(X_train1)
pred_test = mn1.predict(X_test1)
cm=metrics.confusion_matrix(y_tr,pred)
```

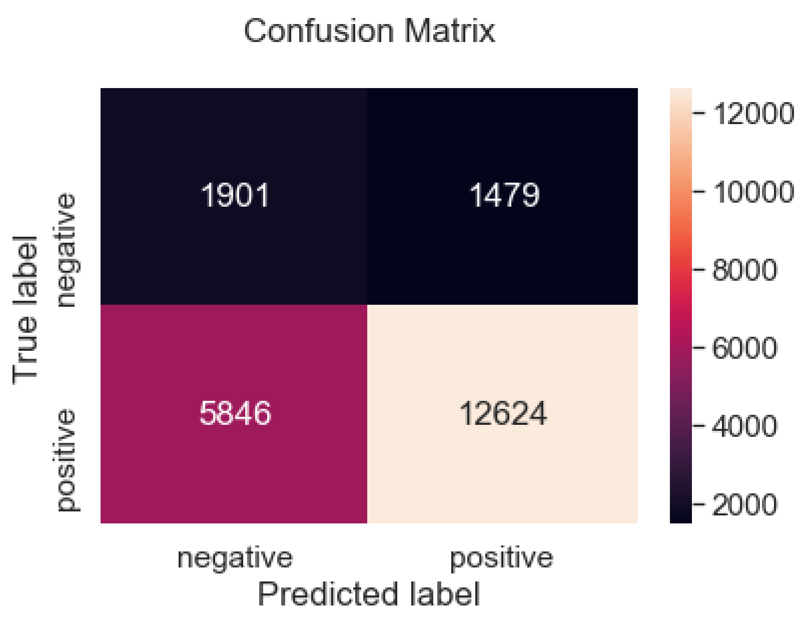
In [86]:
```
ax= plt.subplot()

sns.set(font_scale=1.4)
sns.heatmap(cm,annot=True,ax=ax,fmt='d')

ax.set_xlabel('Predicted label')
ax.set_ylabel('True label')
ax.xaxis.set_ticklabels(['negative','positive'])
ax.yaxis.set_ticklabels(['negative','positive'])
plt.title('Confusion Matrix\n')
```

Out[86]:
```
Text(0.5, 1.0, 'Confusion Matrix\n')
```



In [87]:
```
cm=metrics.confusion_matrix(y_test,pred_test)

ax= plt.subplot()

sns.set(font_scale=1.4)
sns.heatmap(cm,annot=True,ax=ax,fmt='d')

ax.set_xlabel('Predicted label')
ax.set_ylabel('True label')
ax.xaxis.set_ticklabels(['negative','positive'])
ax.yaxis.set_ticklabels(['negative','positive'])
plt.title('Confusion Matrix\n')
```

Out[87]:
```
Text(0.5, 1.0, 'Confusion Matrix\n')
```



3. Conclusions

In [89]:
```
# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vetorizer", "Model", "Hyperparameter", "AUC"]

x.add_row(["BOW", 'Brute', 0.318933436650504550,0.71 ])
x.add_row(["TF_IDF", 'Brute',0.07007623159576255, 0.67])
```

In [90]:
```
print(x)
```

```
+-----------+-------+---------------------+------+
| Vetorizer | Model |    Hyperparameter   | AUC  |
+-----------+-------+---------------------+------+
|    BOW    | Brute | 0.31893343665050455 | 0.71 |
|   TF_IDF  | Brute | 0.07007623159576255 | 0.67 |
+-----------+-------+---------------------+------+
```