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*
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
```

```
F:\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detecte
d Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

In [2]:
```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [3]:
```
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 [4]:
```
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[4]:

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

In [5]:

```
# how to replace elements in list python: https://stackoverflow.com/a/258216
3/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(proje
ct_data.columns)]


#sort dataframe based on time pandas python: https://stackoverflow.com/a/497
02492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_dateti
me'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/1314861
1/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[5]:

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

In [6]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverfl
ow.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-str
ing-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunge
r"
    for j in i.split(','): # it will split it in three parts ["Math & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based
```

```
 on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going
 to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(e
mpty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the t
railing 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]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverfl
ow.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-str
ing-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunge
r"
    for j in i.split(','): # it will split it in three parts ["Math & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based
 on space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going
 to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(e
mpty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the t
railing 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/22898
595/4084039
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 Essay

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

Out[9]:

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

In [10]:
```
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(project_data,project_dat
a['project_is_approved'],stratify=project_data['project_is_approved'],test_s
ize=0.33)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train,stratify=y_
train,test_size=0.33)


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

```
(49041, 18) (49041,)
(24155, 18) (24155,)
(36052, 18) (36052,)
```

In [11]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'su
m'}).reset_index()
X_train = pd.merge(X_train, price_data, on='id', how='left')
X_cv = pd.merge(X_cv, price_data, on='id', how='left')
X_test = pd.merge(X_test, price_data, on='id', how='left')
X_train.head()
```

Out[11]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
0	137784	p189198	dc5ad6f519f577e1c0a8c94d38c80941	Ms.	MI	2 1 0
1	93521	p006442	24eeb5e5c3f9ece7deb7914c2cc70a9e	Mrs.	NC	2 0 1
2	121279	p160880	cec66c2731dde453a0c1b5bd98a92a8b	Mrs.	MI	2 0 1
3	64916	p252782	a2ce65db8c4205dfdfe279622e43b681	Mrs.	NC	2 0 0
4	38319	p093409	d9706f470adf2540b55a488ab6703e1f	Mrs.	FL	2 1 2

In [12]:
```
X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

In [13]:
```
# 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)
```

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classr
```

oom as well as the STEM journals, which my students really enjoyed. I wo
uld love to implement more of the Lakeshore STEM kits in my classroom for
the next school year as they provide excellent and engaging STEM lessons.
My students come from a variety of backgrounds, including language and so
cioeconomic status. Many of them don't have a lot of experience in scien
ce and engineering and these kits give me the materials to provide these
exciting opportunities for my students.Each month I try to do several sci
ence or STEM/STEAM projects. I would use the kits and robot to help guid
e my science instruction in engaging and meaningful ways. I can adapt th
e kits to my current language arts pacing guide where we already teach so
me of the material in the kits like tall tales (Paul Bunyan) or Johnny Ap
pleseed. The following units will be taught in the next school year wher
e I will implement these kits: magnets, motion, sink vs. float, robots.
I often get to these units and don't know If I am teaching the right way
or using the right materials. The kits will give me additional ideas,
strategies, and lessons to prepare my students in science.It is challengi
ng to develop high quality science activities. These kits give me the ma
terials I need to provide my students with science activities that will g
o along with the curriculum in my classroom. Although I have some things
(like magnets) in my classroom, I don't know how to use them effectively.
The kits will provide me with the right amount of materials and show me h
ow to use them in an appropriate way.
==========
I teach high school English to students with learning and behavioral disa
bilities. My students all vary in their ability level. However, the ultim
ate goal is to increase all students literacy levels. This includes their
reading, writing, and communication levels.I teach a really dynamic group
of students. However, my students face a lot of challenges. My students a
ll live in poverty and in a dangerous neighborhood. Despite these challen
ges, I have students who have the the desire to defeat these challenges.
My students all have learning disabilities and currently all are performi
ng below grade level. My students are visual learners and will benefit fr
om a classroom that fulfills their preferred learning style.The materials
I am requesting will allow my students to be prepared for the classroom w
ith the necessary supplies. Too often I am challenged with students who
come to school unprepared for class due to economic challenges. I want m
y students to be able to focus on learning and not how they will be able
to get school supplies. The supplies will last all year. Students will
be able to complete written assignments and maintain a classroom journal.
The chart paper will be used to make learning more visual in class and to
create posters to aid students in their learning. The students have acce
ss to a classroom printer. The toner will be used to print student work
that is completed on the classroom Chromebooks.I want to try and remove a
ll barriers for the students learning and create opportunities for learni
ng. One of the biggest barriers is the students not having the resources
to get pens, paper, and folders. My students will be able to increase the
ir literacy skills because of this project.
==========
\"Life moves pretty fast. If you don't stop and look around once in awhil
e, you could miss it.\" from the movie, Ferris Bueller's Day Off. Think
back...what do you remember about your grandparents? How amazing would i
t be to be able to flip through a book to see a day in their lives?My sec
ond graders are voracious readers! They love to read both fiction and non
fiction books. Their favorite characters include Pete the Cat, Fly Guy,
Piggie and Elephant, and Mercy Watson. They also love to read about insec
ts, space and plants. My students are hungry bookworms! My students are e
ager to learn and read about the world around them. My kids love to be at
school and are like little sponges absorbing everything around them. Thei
r parents work long hours and usually do not see their children. My stude
nts are usually cared for by their grandparents or a family friend. Most
of my students do not have someone who speaks English at home. Thus it is
difficult for my students to acquire language.Now think forward... would

```
n't it mean a lot to your kids, nieces or nephews or grandchildren, to be
able to see a day in your life today 30 years from now? Memories are so p
recious to us and being able to share these memories with future generati
ons will be a rewarding experience.  As part of our social studies curric
ulum, students will be learning about changes over time.  Students will b
e studying photos to learn about how their community has changed over tim
e.  In particular, we will look at photos to study how the land, building
s, clothing, and schools have changed over time.  As a culminating activi
ty, my students will capture a slice of their history and preserve it thr
ough scrap booking. Key important events in their young lives will be doc
umented with the date, location, and names.   Students will be using phot
os from home and from school to create their second grade memories.   The
ir scrap books will preserve their unique stories for future generations
to enjoy.Your donation to this project will provide my second graders wit
h an opportunity to learn about social studies in a fun and creative mann
er.  Through their scrapbooks, children will share their story with other
s and have a historical document for the rest of their lives.
==================================================
\"A person's a person, no matter how small.\" (Dr.Seuss) I teach the smal
lest students with the biggest enthusiasm for learning. My students learn
in many different ways using all of our senses and multiple intelligence
s. I use a wide range of techniques to help all my students succeed. \r\n
Students in my class come from a variety of different backgrounds which m
akes for wonderful sharing of experiences and cultures, including Native
Americans.\r\nOur school is a caring community of successful learners whi
ch can be seen through collaborative student project based learning in an
d out of the classroom. Kindergarteners in my class love to work with han
ds-on materials and have many different opportunities to practice a skill
before it is mastered. Having the social skills to work cooperatively wit
h friends is a crucial aspect of the kindergarten curriculum.Montana is t
he perfect place to learn about agriculture and nutrition. My students lo
ve to role play in our pretend kitchen in the early childhood classroom.
I have had several kids ask me, \"Can we try cooking with REAL food?\" I
will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy
food for snack time. My students will have a grounded appreciation for th
e work that went into making the food and knowledge of where the ingredie
nts came from as well as how it's healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes b
y having us peel our own apples to make homemade applesauce, make our own
bread, and mix up healthy plants from our classroom garden in the spring.
We will also create our own cookbooks to be printed and shared with famil
ies. \r\nStudents will gain math and literature skills as well as a life
long enjoyment for healthy cooking.nannan
==================================================
```

In [14]:

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

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

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

```
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [15]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
\"A person is a person, no matter how small.\" (Dr.Seuss) I teach the sma
llest students with the biggest enthusiasm for learning. My students lear
n in many different ways using all of our senses and multiple intelligenc
es. I use a wide range of techniques to help all my students succeed. \r
\nStudents in my class come from a variety of different backgrounds which
makes for wonderful sharing of experiences and cultures, including Native
Americans.\r\nOur school is a caring community of successful learners whi
ch can be seen through collaborative student project based learning in an
d out of the classroom. Kindergarteners in my class love to work with han
ds-on materials and have many different opportunities to practice a skill
before it is mastered. Having the social skills to work cooperatively wit
h friends is a crucial aspect of the kindergarten curriculum.Montana is t
he perfect place to learn about agriculture and nutrition. My students lo
ve to role play in our pretend kitchen in the early childhood classroom.
I have had several kids ask me, \"Can we try cooking with REAL food?\" I
will take their idea and create \"Common Core Cooking Lessons\" where we
learn important math and writing concepts while cooking delicious healthy
food for snack time. My students will have a grounded appreciation for th
e work that went into making the food and knowledge of where the ingredie
nts came from as well as how it is healthy for their bodies. This project
would expand our learning of nutrition and agricultural cooking recipes b
y having us peel our own apples to make homemade applesauce, make our own
bread, and mix up healthy plants from our classroom garden in the spring.
We will also create our own cookbooks to be printed and shared with famil
ies. \r\nStudents will gain math and literature skills as well as a life
long enjoyment for healthy cooking.nannan
==================================================
```

In [16]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-lin
e-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
 A person is a person, no matter how small.  (Dr.Seuss) I teach the small
est students with the biggest enthusiasm for learning. My students learn
in many different ways using all of our senses and multiple intelligence
s. I use a wide range of techniques to help all my students succeed.   St
udents in my class come from a variety of different backgrounds which mak
es for wonderful sharing of experiences and cultures, including Native Am
ericans.  Our school is a caring community of successful learners which c
an be seen through collaborative student project based learning in and ou
t of the classroom. Kindergarteners in my class love to work with hands-o
n materials and have many different opportunities to practice a skill bef
ore it is mastered. Having the social skills to work cooperatively with f
riends is a crucial aspect of the kindergarten curriculum.Montana is the
perfect place to learn about agriculture and nutrition. My students love
to role play in our pretend kitchen in the early childhood classroom. I h
ave had several kids ask me,  Can we try cooking with REAL food?  I will
take their idea and create  Common Core Cooking Lessons  where we learn i
mportant math and writing concepts while cooking delicious healthy food f
or snack time. My students will have a grounded appreciation for the work
```

```
that went into making the food and knowledge of where the ingredients cam
e from as well as how it is healthy for their bodies. This project would
expand our learning of nutrition and agricultural cooking recipes by havi
ng us peel our own apples to make homemade applesauce, make our own brea
d, and mix up healthy plants from our classroom garden in the spring. We
will also create our own cookbooks to be printed and shared with familie
s.   Students will gain math and literature skills as well as a life long
enjoyment for healthy cooking.nannan
```

In [17]:

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

```
 A person is a person no matter how small Dr Seuss I teach the smallest s
tudents with the biggest enthusiasm for learning My students learn in man
y different ways using all of our senses and multiple intelligences I use
a wide range of techniques to help all my students succeed Students in my
class come from a variety of different backgrounds which makes for wonder
ful sharing of experiences and cultures including Native Americans Our sc
hool is a caring community of successful learners which can be seen throu
gh collaborative student project based learning in and out of the classro
om Kindergarteners in my class love to work with hands on materials and h
ave many different opportunities to practice a skill before it is mastere
d Having the social skills to work cooperatively with friends is a crucia
l aspect of the kindergarten curriculum Montana is the perfect place to l
earn about agriculture and nutrition My students love to role play in our
pretend kitchen in the early childhood classroom I have had several kids
ask me Can we try cooking with REAL food I will take their idea and creat
e Common Core Cooking Lessons where we learn important math and writing c
oncepts while cooking delicious healthy food for snack time My students w
ill have a grounded appreciation for the work that went into making the f
ood and knowledge of where the ingredients came from as well as how it is
healthy for their bodies This project would expand our learning of nutrit
ion and agricultural cooking recipes by having us peel our own apples to
make homemade applesauce make our own bread and mix up healthy plants fro
m our classroom garden in the spring We will also create our own cookbook
s to be printed and shared with families Students will gain math and lite
rature skills as well as a life long enjoyment for healthy cooking nannan
```

In [18]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'yo
u', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'h
e', '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', 'bec
ause', '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', 't
han', 'too', 'very', \
```

```
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shou
ld'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', "sho
uldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

Preprocessing On Train Data :

In [19]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_train = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_train.append(sent.lower().strip())
```

```
100%|██████████| 49041/49041 [00:23<00:00, 2079.74it/s]
```

In [20]:
```
# after preprocesing
preprocessed_essays_train[20000]
```

Out[20]:
```
'celebrate good times come congratulations early intervention program eip
teacher assigned second grade team formerly worked students grades 1 2 5
school title one school sensational second grade class created merging st
udents two classes additional teacher needed serve students second grade
fortunate opportunity work awesome group students diverse group variety t
alents abilities eager learn changed resulted bright eyes warm smiles che
ers tears needless say change not always easily accepted students resilie
nt adapt changes much faster adults however noticed separation two groups
therefore working hard creating caring culture trust collaboration studen
ts graduate ready college career becoming new cohesive happy classroom fa
mily large carpet needed provide comfort personal space student morning m
eetings center time game time daily closings students also need headphone
s five student desktop computers difficult things need starting class nov
ember please allow clear throat good morning good morning good morning se
nsational second graders begin day come together sit carpet begin day pro
blem right carpet small students get fit however small rug not allow much
personal space elbow partners materials requested project help sensationa
l second graders feel good new environment embodies caring trust collabor
ation students feel comfortable learning nannan'
```

Preprocessing On Test Data :

In [21]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_test = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['essay'].values):
```

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

```
100%|██████████| 36052/36052 [00:16<00:00, 2158.07it/s]
```

In [22]:
```
preprocessed_essays_test[10000]
```

Out[22]:
```
'large middle school small town said advantage tight knit community feel
middle school wide variety abilities personalities students coming englis
h classroom sorts different backgrounds abilities able know student whole
student love every minute teaching wether iep learning english second lan
guage gifted talented beyond curriculum normally teach know students leav
ing class authentic learning experience bell rings students pile another
standard boring classroom culture seperated desks old teacher uncomfortab
le not engaged not concentrate lesson fortunately not reality mr beil cla
ss students given choice allowed comfortable learning possibilities endle
ss gotten rid desks created coffee shop feel students given free choice s
eating stand wish not learn isolation community learners want create atmo
sphere promotes continue vision looking purchase standing desk students n
eed move around working would also like purchase comfortable seating stud
ents able concentrate lesson not fact back hurts not stop trying repositi
on lastly would like exercise balls students need movement learning nanna
n'
```

Preprocessing On CV Data :

In [23]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_cv = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_essays_cv.append(sent.lower().strip())
```

```
100%|██████████| 24155/24155 [00:10<00:00, 2199.94it/s]
```

In [24]:
```
preprocessed_essays_cv[10000]
```

Out[24]:
```
'students classes college prep honors level student socio economic status
issue times not students opportunity purchase materials students communit
y supportive cohesive much like large family students interested projects
completing semester receive funding materials need not funding county sch
ool novels materials need use help get students interested ideology socia
l injustice world various cultures history using materials need allow stu
dents explore ideas make social injustices occur well ideas help teach to
lerance avoid injustices ninth grade english students look wide variety l
iterature different areas world way bridge idea social injustice occurs f
requently different cultures decided pair ancient prolific shakespearean
```

```
play one controversial novels twentieth century romeo juliet kill mocking
bird students understanding social injustice happened shakespeare time so
uth 1900 currently occurring allow open discussion create sense community
tolerance based differences among us students need understand society not
always going fair job teach make every experience hope play novel student
s create found social injustice poems use texts use quotes create shirts
using fabric paint students receive credit wearing shirts school discussi
ng students outside class social injustice make sure everyone knows matte
r nannan'
```

1.4 Preprocessing of Project_title Train Data :

In [25]:
```
print(X_train['project_title'].values[0])
print("="*50)
print(X_train['project_title'].values[150])
print("="*50)
print(X_train['project_title'].values[1000])
print("="*50)
print(X_train['project_title'].values[20000])
print("="*50)
```

```
An Apple Today Keeps Illiteracy at Bay
==================================================
Hands on Optics for Young Astronomers
==================================================
You Sit on What?!
==================================================
SELebrate Good Times!     \r\n(Social Emotional Learning)
==================================================
```

In [26]:
```
from tqdm import tqdm
preprocessed_projectitle_train = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['project_title'].values):
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e.lower() not in stopwords)
    preprocessed_projectitle_train.append(sent1.lower().strip())
```

```
100%|██████████| 49041/49041 [00:01<00:00, 44677.17it/s]
```

In [27]:
```
preprocessed_projectitle_train[5000]
```

Out[27]:
```
'putting spin science'
```

1.4 Preprocessing of Project_title CV Data :

In [28]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_projectitle_cv = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['project_title'].values):
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
```

```
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e.lower() not in stopwords)
    preprocessed_projectitle_cv.append(sent1.lower().strip())
```

```
100%|██████████| 24155/24155 [00:00<00:00, 47378.91it/s]
```

In [29]:
```
# after preprocesing
preprocessed_projectitle_cv[19995:20000]
```

Out[29]:
```
['help struggling learners excel using technology supplement',
 'busy bee computer station',
 'steam supplies',
 'reading giving children wings',
 'taking art outside art room']
```

1.4 Preprocessing of Project_title Test Data :

In [30]:
```
from tqdm import tqdm
preprocessed_projectitle_test = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['project_title'].values):
    sent1 = decontracted(sentance)
    sent1 = sent1.replace('\\r', ' ')
    sent1 = sent1.replace('\\"', ' ')
    sent1 = sent1.replace('\\n', ' ')
    sent1 = re.sub('[^A-Za-z0-9]+', ' ', sent1)
    # https://gist.github.com/sebleier/554280
    sent1 = ' '.join(e for e in sent1.split() if e.lower() not in stopwords)
    preprocessed_projectitle_test.append(sent1.lower().strip())
```

```
100%|██████████| 36052/36052 [00:00<00:00, 46349.45it/s]
```

In [31]:
```
preprocessed_projectitle_test[19995:20000]
```

Out[31]:
```
['obstacles wo not get way',
 'bouncing way past distraction learning',
 'additional tools needed foster 21st century learning inquiry',
 'alternative fun comfy supportive seats class',
 'chromebooks learning']
```

DATA PREPROCESSING OF TEACHER_PREFIX IN TRAIN DATA :

In [32]:
```
from tqdm import tqdm
preprocessed_tf_train = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['teacher_prefix'].values):
    sent = sent.replace('\\r', '')
    sent = sent.replace('\\"', '')
    sent = sent.replace('\\n', '')
    sent = re.sub('[^A-Za-z0-9]+', '', sent)
    preprocessed_tf_train.append(sent.lower().strip())
```

```
100%|██████████| 49041/49041 [00:01<00:00, 43925.81it/s]
```

In [33]:
```
X_train['teacher_prefix'].fillna('',inplace=True)
```

DATA PREPROCESSING OF TEACHER_PREFIX IN CV DATA :

In [34]:
```
from tqdm import tqdm
preprocessed_tf_cv = []
# tqdm is for printing the status bar
for sentance in tqdm(X_cv['teacher_prefix'].values):
    sent = sent.replace('\\r', '')
    sent = sent.replace('\\"', '')
    sent = sent.replace('\\n', '')
    sent = re.sub('[^A-Za-z0-9]+', '', sent)
    preprocessed_tf_cv.append(sent.lower().strip())
```

```
100%|██████████| 24155/24155 [00:00<00:00, 42314.60it/s]
```

In [35]:
```
X_cv['teacher_prefix'].fillna('',inplace=True)
```

DATA PREPROCESSING OF TEACHER_PREFIX IN TEST DATA :

In [36]:
```
from tqdm import tqdm
preprocessed_tf_test = []
# tqdm is for printing the status bar
for sentance in tqdm(X_test['teacher_prefix'].values):
    sent = sent.replace('\\r', '')
    sent = sent.replace('\\"', '')
    sent = sent.replace('\\n', '')
    sent = re.sub('[^A-Za-z0-9]+', '', sent)
    preprocessed_tf_test.append(sent.lower().strip())
```

```
100%|██████████| 36052/36052 [00:00<00:00, 43820.61it/s]
```

In [37]:
```
X_test['teacher_prefix'].fillna('',inplace=True)
```

1.5 Preparing Data For Models

In [38]:
```
project_data.columns
```

Out[38]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_stat
e',
       'Date', 'project_grade_category', 'project_title', 'project_essay_
1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approv
ed',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

ONE HOT ENCODING OF CLEAN_CATAGORIES IN TRAIN,TEST,CV DATA :

In [39]:
```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizerc = CountVectorizer()
vectorizerc.fit(X_train['clean_categories'].values)
categories_one_hot_train = vectorizerc.transform(X_train['clean_categories']
.values)
categories_one_hot_test = vectorizerc.transform(X_test['clean_categories'].v
alues)
categories_one_hot_cv = vectorizerc.transform(X_cv['clean_categories'].value
s)
print(vectorizerc.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",categories_one
_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",categories_one_
hot_test.shape)
print("Shape of matrix of CV data after one hot encoding ",categories_one_ho
t_cv.shape)
```

```
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'li
teracy_language', 'math_science', 'music_arts', 'specialneeds', 'warmth']
Shape of matrix of Train data after one hot encoding  (49041, 9)
Shape of matrix of Test data after one hot encoding  (36052, 9)
Shape of matrix of CV data after one hot encoding  (24155, 9)
```

ONE HOT ENCODING OF CLEAN_SUB_CATAGORIES IN TRAIN,TEST,CV DATA :

In [40]:
```
# we use count vectorizer to convert the values into one
vectorizersc = CountVectorizer()
vectorizersc.fit(X_train['clean_subcategories'].values)
```

```
sub_categories_one_hot_train = vectorizersc.transform(X_train['clean_subcate
gories'].values)
sub_categories_one_hot_test = vectorizersc.transform(X_test['clean_subcatego
ries'].values)
sub_categories_one_hot_cv = vectorizersc.transform(X_cv['clean_subcategorie
s'].values)
print(vectorizersc.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",sub_categories
_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",sub_categories_
one_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding ",sub
_categories_one_hot_cv.shape)
```

```
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_governme
nt', 'college_careerprep', 'communityservice', 'earlydevelopment', 'econo
mics', 'environmentalscience', 'esl', 'extracurricular', 'financiallitera
cy', 'foreignlanguages', 'gym_fitness', 'health_lifescience', 'health_wel
lness', 'history_geography', 'literacy', 'literature_writing', 'mathemati
cs', 'music', 'nutritioneducation', 'other', 'parentinvolvement', 'perfor
mingarts', 'socialsciences', 'specialneeds', 'teamsports', 'visualarts',
'warmth']
Shape of matrix of Train data after one hot encoding  (49041, 30)
Shape of matrix of Test data after one hot encoding  (36052, 30)
Shape of matrix of Cross Validation data after one hot encoding  (24155,
30)
```

ONE HOT ENCODING OF SCHOOL STATE IN TEST,TRAIN,CV DATA :

In [41]:
```
vectorizerss = CountVectorizer()
vectorizerss.fit(X_train['school_state'].values)
school_state_categories_one_hot_train = vectorizerss.transform(X_train['scho
ol_state'].values)
school_state_categories_one_hot_test = vectorizerss.transform(X_test['school
_state'].values)
school_state_categories_one_hot_cv = vectorizerss.transform(X_cv['school_sta
te'].values)
print(vectorizerss.get_feature_names())
print("Shape of matrix of Train data after one hot encoding",school_state_ca
tegories_one_hot_train.shape)
print("Shape of matrix of Test data after one hot encoding ",school_state_ca
tegories_one_hot_test.shape)
print("Shape of matrix of Cross Validation data after one hot encoding",scho
ol_state_categories_one_hot_cv.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi',
'ia', 'id', 'il', 'in', 'ks', '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 of Train data after one hot encoding (49041, 51)
Shape of matrix of Test data after one hot encoding  (36052, 51)
Shape of matrix of Cross Validation data after one hot encoding (24155, 5
1)
```

ONE HOT ENCODING OF TEACHER PREFIX IN TEST,TRAIN,CV DATA :

In [42]:
```
#Teacher Prefix
vectorizertp = CountVectorizer()
tp_one_hot=vectorizertp.fit(X_train['teacher_prefix'].values)
teacher_prefix_categories_one_hot_train =vectorizertp.transform(X_train['tea
cher_prefix'].values)
teacher_prefix_categories_one_hot_test =vectorizertp.transform(X_test['teach
er_prefix'].values)
teacher_prefix_categories_one_hot_cv=vectorizertp.transform(X_cv['teacher_pr
efix'].values)
print(vectorizertp.get_feature_names())
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_on
e_hot_train.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_on
e_hot_test.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_on
e_hot_cv.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix after one hot encoding  (49041, 5)
Shape of matrix after one hot encoding  (36052, 5)
Shape of matrix after one hot encoding  (24155, 5)
```

ONE HOT ENCODING OF PROJECT GRADE CATAGORY IN TEST,TRAIN,CV DATA:

In [43]:
```
from tqdm import tqdm
preprocessed_pg_train = []
# tqdm is for printing the status bar
for sent in tqdm(X_train['project_grade_category'].values):
    s=[]
    s=sent.split(" ")
    s[0]=s[0].replace("Grades","Grades_")
    sent=("").join(s)
    preprocessed_pg_train.append(sent.lower().strip())
from tqdm import tqdm
preprocessed_pg_cv = []
# tqdm is for printing the status bar
for sent in tqdm(X_cv['project_grade_category'].values):
    s=[]
    s=sent.split(" ")
    s[0]=s[0].replace("Grades","Grades_")
    sent=("").join(s)
    preprocessed_pg_cv.append(sent.lower().strip())
from tqdm import tqdm
preprocessed_pg_test = []
# tqdm is for printing the status bar
for sent in tqdm(X_test['project_grade_category'].values):
    s=[]
    s=sent.split(" ")
    s[0]=s[0].replace("Grades","Grades_")
    sent=("").join(s)
    preprocessed_pg_test.append(sent.lower().strip())
```

```
100%|██████████| 49041/49041 [00:00<00:00, 723135.01it/s]
100%|██████████| 24155/24155 [00:00<00:00, 733974.33it/s]

100%|██████████| 36052/36052 [00:00<00:00, 753139.29it/s]
```

In [44]:
```
set(preprocessed_pg_train)
```

Out[44]: {'grades_3-5', 'grades_6-8', 'grades_9-12', 'grades_prek-2'}

In [45]:
```
vectorizerpg = CountVectorizer(vocabulary=set(preprocessed_pg_train))
vectorizerpg.fit(set(preprocessed_pg_train))
print(vectorizerpg.get_feature_names())
pgc_one_hot_train=vectorizerpg.transform(preprocessed_pg_train)
pgc_one_hot_cv=vectorizerpg.transform(preprocessed_pg_cv)
pgc_one_hot_test=vectorizerpg.transform(preprocessed_pg_test)
print("Shape of matrix after one hot encoding ",pgc_one_hot_train.shape)
print("Shape of matrix after one hot encoding ",pgc_one_hot_cv.shape)
print("Shape of matrix after one hot encoding ",pgc_one_hot_test.shape)
```

```
['grades_3-5', 'grades_6-8', 'grades_9-12', 'grades_prek-2']
Shape of matrix after one hot encoding  (49041, 4)
Shape of matrix after one hot encoding  (24155, 4)
Shape of matrix after one hot encoding  (36052, 4)
```

1.5.2 Vectorizing Text data

Bag of words on ESSAY in TRAIN , TEST AND CV Data :

In [46]:
```
# We are considering only the words which appeared in at least 10 documents
(rows or projects).
vectorizerbowe = CountVectorizer(min_df=10)
vectorizerbowe.fit(preprocessed_essays_train)
text_bow_train = vectorizerbowe.transform(preprocessed_essays_train)
text_bow_cv = vectorizerbowe.transform(preprocessed_essays_cv)
text_bow_test = vectorizerbowe.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_bow_train.shape)
print("Shape of matrix after one hot encoding ",text_bow_cv.shape)
print("Shape of matrix after one hot encoding ",text_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (49041, 12015)
Shape of matrix after one hot encoding  (24155, 12015)
Shape of matrix after one hot encoding  (36052, 12015)
```

Bag of words on TITLE in TRAIN , TEST AND CV Data :

In [47]:
```
vectorizerbowt = CountVectorizer(min_df=10)
vectorizerbowt.fit(preprocessed_projectitle_train)
title_bow_train = vectorizerbowt.transform(preprocessed_projectitle_train)
title_bow_cv = vectorizerbowt.transform(preprocessed_projectitle_cv)
title_bow_test = vectorizerbowt.transform(preprocessed_projectitle_test)
print("Shape of matrix after one hot encoding ",title_bow_train.shape)
print("Shape of matrix after one hot encoding ",title_bow_cv.shape)
print("Shape of matrix after one hot encoding ",title_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (49041, 1991)
Shape of matrix after one hot encoding  (24155, 1991)
Shape of matrix after one hot encoding  (36052, 1991)
```

TFIDF vectorizer on ESSAY in TRAIN , CV & TEST DATA :

In [48]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizertie = TfidfVectorizer(min_df=10)
```

```
vectorizertie.fit(preprocessed_essays_train)
text_tfidf_train = vectorizertie.transform(preprocessed_essays_train)
text_tfidf_cv = vectorizertie.transform(preprocessed_essays_cv)
text_tfidf_test = vectorizertie.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_tfidf_train.shape)
print("Shape of matrix after one hot encoding ",text_tfidf_cv.shape)
print("Shape of matrix after one hot encoding ",text_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (49041, 12015)
Shape of matrix after one hot encoding  (24155, 12015)
Shape of matrix after one hot encoding  (36052, 12015)
```

TFIDF vectorizer on TITLE in TRAIN , CV & TEST DATA :

In [49]:
```
vectorizertit = TfidfVectorizer(min_df=10)
vectorizertit.fit(preprocessed_projectitle_train)
title_tfidf_train = vectorizertit.transform(preprocessed_projectitle_train)
title_tfidf_cv = vectorizertit.transform(preprocessed_projectitle_cv)
title_tfidf_test = vectorizertit.transform(preprocessed_projectitle_test)
print("Shape of matrix after one hot encoding ",title_tfidf_train.shape)
print("Shape of matrix after one hot encoding ",title_tfidf_cv.shape)
print("Shape of matrix after one hot encoding ",title_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (49041, 1991)
Shape of matrix after one hot encoding  (24155, 1991)
Shape of matrix after one hot encoding  (36052, 1991)
```

Vectorizing Numerical features

1) PRICE

In [50]:
```
from sklearn.preprocessing import MinMaxScaler
from sklearn import preprocessing
price_scalar = MinMaxScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1))
price_train= price_scalar.transform(X_train['price'].values.reshape(-1, 1))
price_test= price_scalar.transform(X_test['price'].values.reshape(-1, 1))
price_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
print(price_train.shape, y_train.shape)
print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
```

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

In [51]:
```
print("The shape of training is",price_train.shape, y_train.shape)
print("The shape of cv is",price_cv.shape, y_cv.shape)
print("The shape of test is",price_test.shape, y_test.shape)
```

```
The shape of training is (49041, 1) (49041,)
The shape of cv is (24155, 1) (24155,)
The shape of test is (36052, 1) (36052,)
```

2) Quantity :

In [52]:
```
quantity_scaler = MinMaxScaler()
quantity_scaler.fit(X_train['quantity'].values.reshape(-1,1))
quantity_train = quantity_scaler.transform(X_train['quantity'].values.reshap
e(-1,1))
quantity_cv = quantity_scaler.transform(X_cv['quantity'].values.reshape(-1,1
))
quantity_test = quantity_scaler.transform(X_test['quantity'].values.reshape(
-1,1))
print("After vectorization")
print(quantity_train.shape, y_train.shape)
print(quantity_cv.shape, y_cv.shape)
print(quantity_test.shape, y_test.shape)
```

```
F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConve
rsionWarning:

Data with input dtype int64 was converted to float64 by MinMaxScaler.
```

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

In [53]:
```
print("The shape of training is",quantity_train.shape, y_train.shape)
print("The shape of cv is",quantity_cv.shape, y_cv.shape)
print("The shape of test is",quantity_test.shape, y_test.shape)
```

```
The shape of training is (49041, 1) (49041,)
The shape of cv is (24155, 1) (24155,)
The shape of test is (36052, 1) (36052,)
```

3) Number Of Projects Proposed By Teachers :

In [54]:
```
noofprojects = MinMaxScaler()
noofprojects.fit(X_train['teacher_number_of_previously_posted_projects'].val
ues.reshape(-1,1))
prev_projects_train = noofprojects.transform(X_train['teacher_number_of_prev
iously_posted_projects'].values.reshape(-1,1))
prev_projects_cv = noofprojects.transform(X_cv['teacher_number_of_previously
_posted_projects'].values.reshape(-1,1))
prev_projects_test = noofprojects.transform(X_test['teacher_number_of_previo
usly_posted_projects'].values.reshape(-1,1))
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
```

```
F:\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConve
rsionWarning:

Data with input dtype int64 was converted to float64 by MinMaxScaler.
```

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

Merging all the above features

SET 1 :

In [55]:
```
from scipy.sparse import hstack
X_tr = hstack((text_bow_train,title_bow_train,school_state_categories_one_ho
t_train,pgc_one_hot_train,teacher_prefix_categories_one_hot_train,categories
_one_hot_train,sub_categories_one_hot_train,prev_projects_train,price_train,
quantity_train)).tocsr()
X_cr = hstack((text_bow_cv,title_bow_cv,school_state_categories_one_hot_cv,p
gc_one_hot_cv,teacher_prefix_categories_one_hot_cv,categories_one_hot_cv,sub
_categories_one_hot_cv,prev_projects_cv,price_cv,quantity_cv)).tocsr()
X_te = hstack((text_bow_test,title_bow_test,school_state_categories_one_hot_
test,pgc_one_hot_test,teacher_prefix_categories_one_hot_test,categories_one_
hot_test,sub_categories_one_hot_test,prev_projects_test,price_test,quantity_
test)).tocsr()

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

```
Final Data matrix
(49041, 14108) (49041,)
(24155, 14108) (24155,)
(36052, 14108) (36052,)
```

SET 2 :

In [56]:
```
X_tr2 = hstack((title_tfidf_train,text_tfidf_train,school_state_categories_o
ne_hot_train,pgc_one_hot_train,teacher_prefix_categories_one_hot_train,categ
ories_one_hot_train,sub_categories_one_hot_train,prev_projects_train,price_t
rain,quantity_train)).tocsr()
X_cr2 = hstack((title_tfidf_cv,text_tfidf_cv,school_state_categories_one_hot
_cv,pgc_one_hot_cv,teacher_prefix_categories_one_hot_cv,categories_one_hot_c
v,sub_categories_one_hot_cv,prev_projects_cv,price_cv,quantity_cv)).tocsr()
X_te2 = hstack((title_tfidf_test,text_tfidf_test,school_state_categories_one
_hot_test,pgc_one_hot_test,teacher_prefix_categories_one_hot_test,categories
_one_hot_test,sub_categories_one_hot_test,prev_projects_test,price_test,quan
tity_test)).tocsr()

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

```
Final Data matrix
(49041, 14108) (49041,)
(24155, 14108) (24155,)
(36052, 14108) (36052,)
```

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 values of `feature_log_prob_` 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

2. Naive Bayes

2.4 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

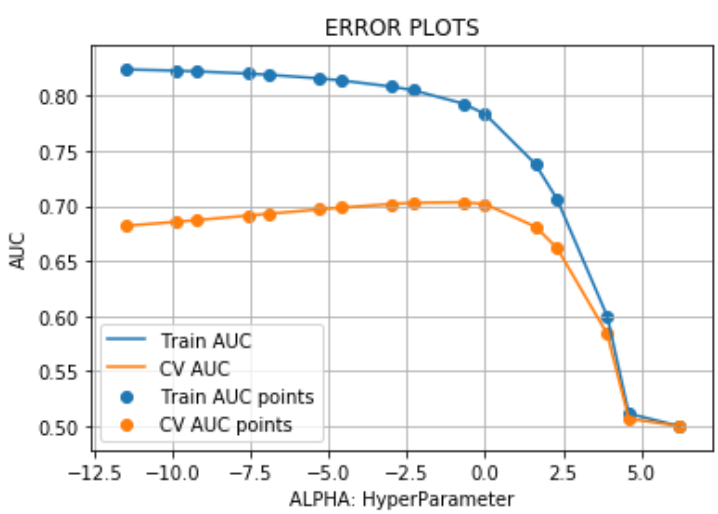
In [57]:
```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score

train_auc = []
```

```
cv_auc = []
log_alphas =[]

alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1,
0.5, 1, 5, 10, 50, 100, 500]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior = [0.5,0.5])
    nb.fit(X_tr, y_train)
    y_train_pred = nb.predict_proba(X_tr)[:,1]
    y_cv_pred = nb.predict_proba(X_cr)[:,1]
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

```
100%|██████████| 16/16 [00:01<00:00, 12.33it/s]
```

In [58]:

```
import numpy
plt.plot(numpy.log(alphas), train_auc, label='Train AUC')
plt.plot(numpy.log(alphas), cv_auc, label='CV AUC')

plt.scatter(numpy.log(alphas), train_auc, label='Train AUC points')
plt.scatter(numpy.log(alphas), cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("ALPHA: HyperParameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



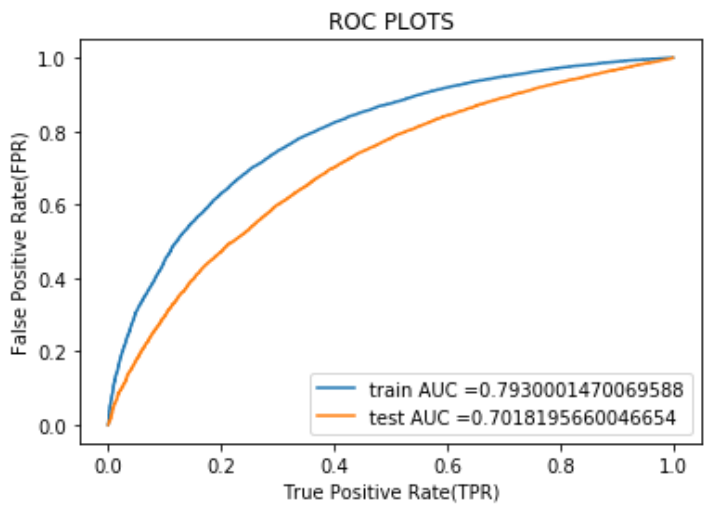
In [59]:

```
best_alpha = 0.5
```

In [60]:

```
from sklearn.metrics import roc_curve, auc
neigh = MultinomialNB(alpha=best_alpha,class_prior = [0.5,0.5])
neigh.fit(X_tr,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates of the positive class
# not the predicted outputs
train_fpr, train_tpr, thresholds = roc_curve(y_train, neigh.predict_proba(X_
tr)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, neigh.predict_proba(X_te)
```

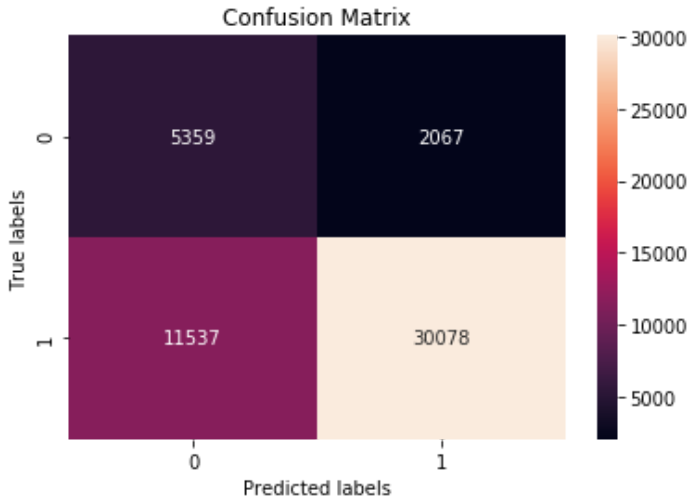
```
[:,1])
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_
tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr
)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC PLOTS")
plt.show()
print("="*100)
```



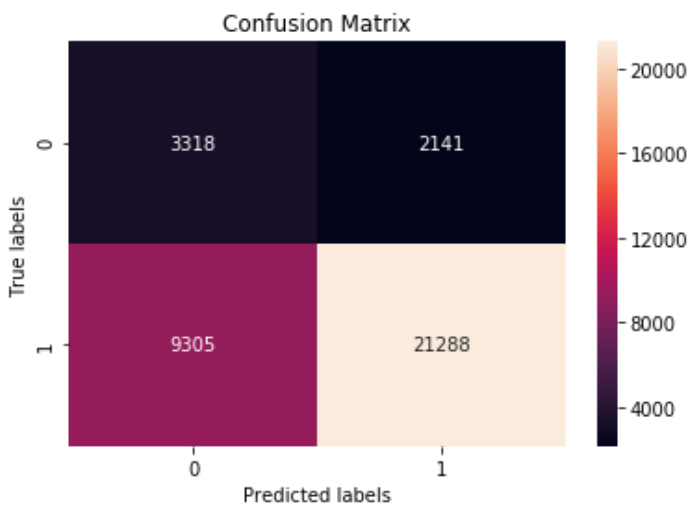

```
==================================================================================
==========================
```

Confusion Matrix

In [61]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-mat
rix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_train, neigh.predict(X_tr)), 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');
```



In [62]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-mat
rix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, neigh.predict(X_te)), 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');
```



In [66]:
```
#for BOW
nb = MultinomialNB(alpha=0.5,class_prior = [0.5,0.5])# takes the k from the
 i th list value
nb.fit(X_tr,y_train)# fit the model
# now make a dictionary of all the probabilities fo the weights
bow_features_names = []
for a in vectorizerbowe.get_feature_names(): # essays bow
    bow_features_names.append(a)
for a in vectorizerbowt.get_feature_names(): #titles bow
    bow_features_names.append(a)
for a in vectorizerss.get_feature_names() :#school state
    bow_features_names.append(a)
for a in vectorizerpg.get_feature_names() :# project categories
    bow_features_names.append(a)
```

```
for a in vectorizertp.get_feature_names() :# teacher prefix
    bow_features_names.append(a)
for a in vectorizerc.get_feature_names() :# clean categories
    bow_features_names.append(a)
for a in vectorizersc.get_feature_names() :# sub categoreis
    bow_features_names.append(a)
bow_features_names.extend(['prev_projects_train','price_train','quantity_tra
in'])
print(len(bow_features_names))
```

```
14108
14108
```

In [71]:
```
bow_features_names[-4]
```

Out[71]: 'warmth'

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

In [72]:
```
neg_class_prob_sorted = neigh.feature_log_prob_[0, :].argsort()
pos_class_prob_sorted = neigh.feature_log_prob_[1, :].argsort()
print(np.take(bow_features_names, neg_class_prob_sorted[:10]))
print(np.take(bow_features_names, pos_class_prob_sorted[:10]))
```

```
['humidity' 'emerging' 'pendulum' 'deconstruct' 'embracing' 'centerpiece'
 'targets' 'tribulations' 'goodall' 'karaoke']
['grades_prek-2' 'grades_9-12' 'grades_6-8' 'grades_3-5' 'dr' 'chefs'
 'seed' 'tad' 'rosebud' 'investigators']
```

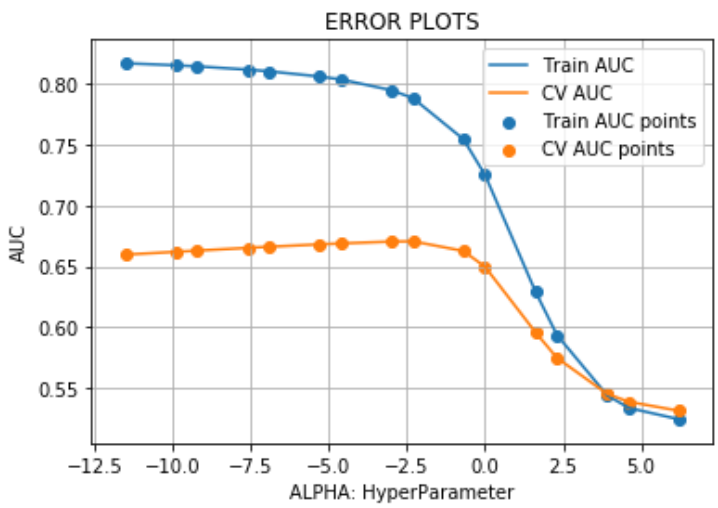
2.4.2 Applying Naive Bayes on TFIDF, SET 2

In [73]:
```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score

train_auc = []
cv_auc = []
log_alphas =[]

alphas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1,
0.5, 1, 5, 10, 50, 100, 500]

for i in tqdm(alphas):
    nb = MultinomialNB(alpha = i,class_prior = [0.5,0.5])
    nb.fit(X_tr2, y_train)
    y_train_pred = nb.predict_proba(X_tr2)[:,1]
    y_cv_pred = nb.predict_proba(X_cr2)[:,1]
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
estimates of the positive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

```
100%|██████████| 16/16 [00:01<00:00, 12.43it/s]
```

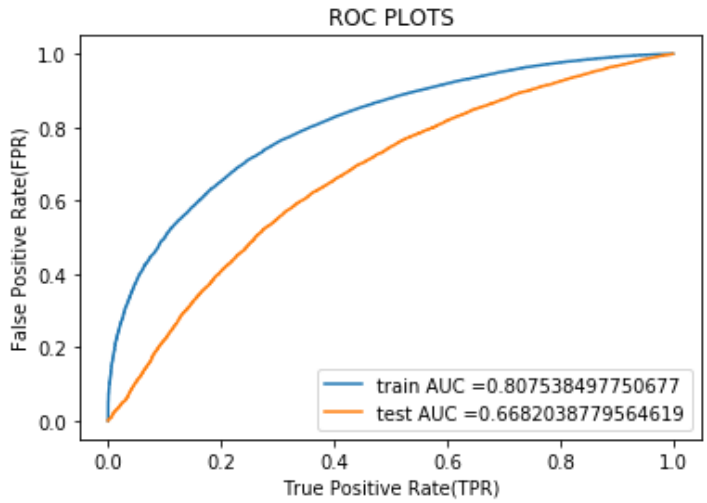
In [74]:
```
import numpy
plt.plot(numpy.log(alphas), train_auc, label='Train AUC')
```

```
plt.plot(numpy.log(alphas), cv_auc, label='CV AUC')

plt.scatter(numpy.log(alphas), train_auc, label='Train AUC points')
plt.scatter(numpy.log(alphas), cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("ALPHA: HyperParameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [75]:
```
best_alpha = 0.005
```

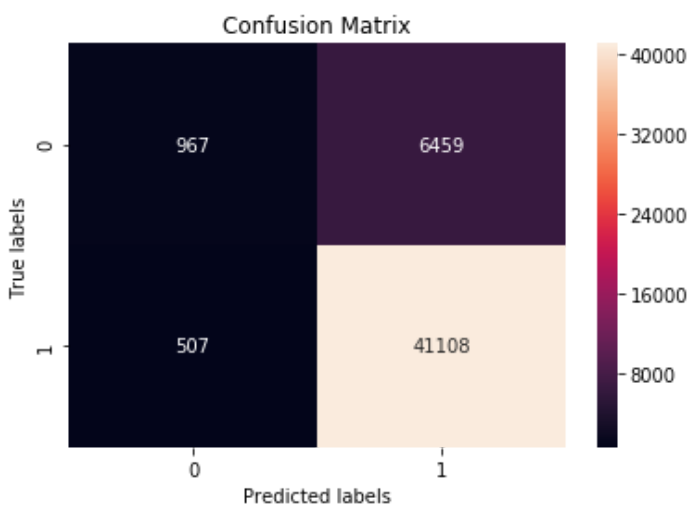
In [76]:
```
from sklearn.metrics import roc_curve, auc
neigh = MultinomialNB(alpha=0.003)
neigh.fit(X_tr2,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates of the positive class
# not the predicted outputs
train_fpr, train_tpr, thresholds = roc_curve(y_train, neigh.predict_proba(X_
tr2)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, neigh.predict_proba(X_te2
)[:,1])
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_
tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr
)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC PLOTS")
plt.show()
print("="*100)
```



==========
==========

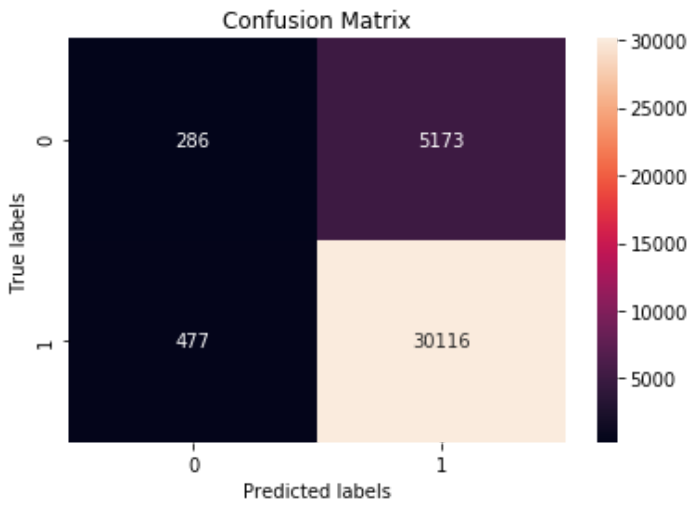
Confusion Matrix Of SET 2 :

In [77]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-mat
rix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_train, neigh.predict(X_tr2)), 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');
```



In [78]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-mat
rix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, neigh.predict(X_te2)), 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');
```



In [79]:
```
#for BOW
nb = MultinomialNB(alpha=0.005,class_prior = [0.5,0.5])# takes the k from th
e i th list value
nb.fit(X_tr2,y_train)# fit the model
# now make a dictionary of all the probabilities fo the weights
tf_features_names = []
for a in vectorizertie.get_feature_names(): # essays bow
    tf_features_names.append(a)
for a in vectorizertit.get_feature_names(): #titles bow
    tf_features_names.append(a)
for a in vectorizerss.get_feature_names() :#school state
    tf_features_names.append(a)
for a in vectorizerpg.get_feature_names() :# project categories
    tf_features_names.append(a)
for a in vectorizertp.get_feature_names() :# teacher prefix
    tf_features_names.append(a)
for a in vectorizerc.get_feature_names() :# clean categories
    tf_features_names.append(a)
for a in vectorizersc.get_feature_names() :# sub categoreis
    tf_features_names.append(a)
tf_features_names.extend(['prev_projects_train','price_train','quantity_trai
n'])
print(len(tf_features_names))
```

```
14108
```

In [80]:
```
print( len(tf_features_names))
```

```
14108
```

Top 10 important features of positive and negative class :

In [81]:
```
neg_class_prob_sorted = neigh.feature_log_prob_[0, :].argsort()
pos_class_prob_sorted = neigh.feature_log_prob_[1, :].argsort()
print(np.take(tf_features_names, neg_class_prob_sorted[:10]))
print(np.take(tf_features_names, pos_class_prob_sorted[:10]))
```

```
['myths' 'seedlings' 'seem' 'sei' 'buzz' 'selecting' 'selfless'
 'cafeteria' 'sends' 'september']
['grades_3-5' 'grades_prek-2' 'grades_9-12' 'grades_6-8' 'react'
 'entrepreneurs' 'nye' 'discourse' 'tale' 'twinkle']
```

3. Conclusions

In [84]:

```python
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]

x.add_row(["BOW","Auto",0.5,71])
x.add_row(["Tf-Idf","Auto",0.005,67])
print(x.get_string(titles = "KNN - Observations"))
```

```
+------------+-------+-----------------+-----+
| Vectorizer | Model | Hyper Parameter | AUC |
+------------+-------+-----------------+-----+
|    BOW     |  Auto |       0.5       |  71 |
|   Tf-Idf   |  Auto |      0.005      |  67 |
+------------+-------+-----------------+-----+
```