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

1.2 Preprocessing of Project Subject Categories :

In [5]:
```
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 [6]:
```
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 [7]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str)+project_dat
a["project_essay_2"].map(str)+project_data["project_essay_3"].map(str)+proje
ct_data["project_essay_4"].map(str)
```

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

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2(
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2(

Train And Test Data Split

In [9]:
```
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 [10]:
```
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[10]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	p
0	141127	p251147	e0228a25ff1f48a9c53efaa16147c6de	Ms.	TX	2
1	179146	p257031	376c278c3eb7321c7553c7dbadebdc96	Mrs.	MA	2
2	114125	p237493	d6557d984c482a4cb99b452c8e915242	Ms.	CA	2

3	103886	p055883	9453b73c0834ab9f5a35bab0734d5a9c	Ms.	NY	2
4	154510	p106383	c9db811b2da0f01d5d5384a9af29ce67	Ms.	CT	2

In [11]:
```
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 [12]:
```
# printing some random reviews
print(X_train['essay'].values[0])
print("="*50)
print(X_train['essay'].values[150])
print("="*50)
print(X_train['essay'].values[1000])
print("="*50)
print(X_train['essay'].values[20000])
print("="*50)
```

```
My students come from a low social economic background with very little r
esources and technology. We together attend a Title I school. All the stu
dents strive for excellence using the resources that we have but are ofte
n limited when it comes to enrichment and research activities. \r\nThese
students come from broken families. Many of their parents are not involve
d in the child's life and grandparents are now guardians. We even have ha
d students whose parents passed away this year and students whose sibling
s have been separated. \r\nThese students have many many educational dist
ractions and require multiple behavior and study plans.\r\nThese students
deserve the opportunity to reach their full potential with all the tools
along with doing so.These I Pads are a necessity in the this generation o
f students. They must learn to be skilled on these devices. The students
can use these to work on education galaxy (an online educational progra
m), expanding their knowledge in history on research projects and even in
enrichment activities to discover various knowledge that they would not h
ave previously been able to reach without the use of technology. There ar
e many supporting apps and systems that can go along with these devices t
hat all aid in the educational experience. With the use of technology on
the I Pad, we are able to reach a wider variety of students and meet all
of their educational needs. These devices also allow us to teach for diff
erent learning styles. \r\nThis project will make a difference for the th
ese students by simply giving them the educational opportunities that the
y otherwise have not been exposed to but greatly deserve.nannan
==================================================
My students come from all walks of life, they have different backgrounds,
```

face different challenges, but most importantly they are determined to su
cceed. I teach at large Title I school in North Florida, our students inc
luded general education students, special need students, English Language
Learners (ELL), and gifted and talented students. Our school is the only
technology magnet school for students in grades K-5. As a technology magn
et, we focus on providing a safe a nurturing learning environment while i
ncreasing student's access to technology. \r\n In many cases the only
access our students have to technology is on campus; I want our classroom
to be a place where students can learn and design technology that impact
their lives. Majority of my students reside in the neighboring housing
community, although their access to resources are limited their will and
determination to succeed is limitless. My students have so much to worry
about in their personal lives, therefore I want my classroom to be a plac
e where students can escape, relax, and feel safe. By removing resource b
arriers in our classroom my students can focus on their academics includi
ng learning about new technological advancements.\r\nMy students live in
a digital world, where smart phones, text messaging, and social media dri
ves so much of what they see and learn. Many of our local middle and high
schools provide students with opportunities to explore STEM (Science, Tec
hnology, Engineering, and Mathematics) by offering classes and clubs that
focus on STEM related issues. I want my students to be able to explore,
observe, and discover the world around them through hands-on activities,
which allow them to see how scientist solve day-to-day problems.\r\n\r\n
I want to provide meaningful learning opportunities for my students, I be
lieve by exposing my students to STEM initiatives at a young age, I can h
elp them discover a love for math and science. Working in small groups,
students will use the mini iPads and apps included with The Wonder Worksh
op Dash Robot to learn about programming. During small groups activities
students will create programs that will allow Dash the Robot to dance, mo
ve, and react to students’ voice commands. In addition, students will be
able to program the Dash the Dot Xylophone to create music and design spe
ctacular lightshows.\r\n\r\n This project will provide my students w
ith numerous opportunities to be innovating and creative. With over 100 c
oding adventures my students will have multiple chances to explore progra
mming with Dash the Robots. The coding exercises will also help my studen
ts solve and create real world math problems. When you support this proje
ct, you are providing my students countless opportunities to enhance thei
r understanding of technology. When Programming and Robotics Collide, my
kids will experience technology in a new and meaningful way.\r\nnannan
==========
A typical day in my classroom begins with many smiles and hugs from my st
udents. We teach and learn from bell to bell, and bright ideas are always
blooming in their little minds. I want students to have every opportunity
to learn everything that I teach them no matter what their circumstances
are.I teach the most fabulous group of second graders in Mississippi. My
school is 95% African American students and 100% free meals. Parent invol
vement is next to nonexistent. Teachers at my school wear several differe
nt hats throughout the day. My students love giving hugs and learning. Th
ey are bursting with smiles and stories from morning to afternoon, and th
ere is never a dull moment in my classroom. I don't teach in a \"quiet\"
room. It is my belief that there is no learning going on in a \"quiet\" r
oom. I believe that students should be able to express themselves verball
y as much as possible and also interact with classmates. I want students
to be able to communicate with other second graders in our state to see w
hat's going on in their classrooms.The materials that I am requesting wil
l play a major role in helping my class meet the task of communicating wi
th other second graders in our state about what's going on in our school
and finding out what other students are learning. We will work on skills
that they've learned through writing letters to work on things such as us
ing correct punctuation, capitalization, and the four types of sentences.
We will also incorporate holiday themes, create booklets, and write autob
iographies to send to pen-pals.Donations for this project will help make

```
a tremendous difference in what my students can do in the classroom. Many
students do not have the bare necessities from day one. I spend all of my
school funding and at least another 400 dollars out-of-pocket to make sur
e that students have what they need every day. When students have what th
ey need to function, it boosts their confidence levels and takes away the
worry of simply not having a pencil when the reason behind it is not so s
imple at all.
==================================================
My students come from various backgrounds in a high need urban school dis
trict. Working in the Fruitvale area of Oakland, we have a high percentag
e of English Language Learners. \r\n\r\nHowever, all learners, no matter
their background, can achieve and should be given the support that they n
eed and deserve. Our school uses Waldorf-inspired teaching methods to hel
p target instruction to all learners of our diverse community. \r\n\r\nMy
students thrive when they are actively engaged and taking charge of their
learning. When given the opportunity to investigate beyond books and vide
os I have the opportunity to see my students work cooperatively, think cr
itically, and let their creativity soar.English is a second language for
many of my students. As research and experience tells us, visuals are an
essential part of quality teaching anywhere, but particularly English Lan
guage Learners. By providing my students with this document camera, you w
ill be giving them the opportunity to view multiple solutions to math pro
blems, see other students' work, justify answers, learn effective note ta
king skills, complete graphic organizers, learn to use a variety of math
manipulatives and view real life objects, compose classwide pieces of wri
ting, see writing modeled, edit text... and the list goes on! My students
understand better and are able to take content to a higher level when the
y have a reference in front of them. With a document camera, I will be ab
le to present and model new material in a variety of ways to meet the nee
ds of all my studentsnannan
==================================================
```

In [13]:
```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 [14]:
```python
sent = decontracted(X_train['essay'].values[20000])
print(sent)
print("="*50)
```

```
My students come from various backgrounds in a high need urban school dis
trict. Working in the Fruitvale area of Oakland, we have a high percentag
e of English Language Learners. \r\n\r\nHowever, all learners, no matter
their background, can achieve and should be given the support that they n
eed and deserve. Our school uses Waldorf-inspired teaching methods to hel
p target instruction to all learners of our diverse community. \r\n\r\nMy
```

```
students thrive when they are actively engaged and taking charge of their
learning. When given the opportunity to investigate beyond books and vide
os I have the opportunity to see my students work cooperatively, think cr
itically, and let their creativity soar.English is a second language for
many of my students. As research and experience tells us, visuals are an
essential part of quality teaching anywhere, but particularly English Lan
guage Learners. By providing my students with this document camera, you w
ill be giving them the opportunity to view multiple solutions to math pro
blems, see other students' work, justify answers, learn effective note ta
king skills, complete graphic organizers, learn to use a variety of math
manipulatives and view real life objects, compose classwide pieces of wri
ting, see writing modeled, edit text... and the list goes on! My students
understand better and are able to take content to a higher level when the
y have a reference in front of them. With a document camera, I will be ab
le to present and model new material in a variety of ways to meet the nee
ds of all my studentsnannan
==================================================
```

In [15]:
```
# \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)
```

```
My students come from various backgrounds in a high need urban school dis
trict. Working in the Fruitvale area of Oakland, we have a high percentag
e of English Language Learners.     However, all learners, no matter thei
r background, can achieve and should be given the support that they need
and deserve. Our school uses Waldorf-inspired teaching methods to help ta
rget instruction to all learners of our diverse community.     My student
s thrive when they are actively engaged and taking charge of their learni
ng. When given the opportunity to investigate beyond books and videos I h
ave the opportunity to see my students work cooperatively, think critical
ly, and let their creativity soar.English is a second language for many o
f my students. As research and experience tells us, visuals are an essent
ial part of quality teaching anywhere, but particularly English Language
Learners. By providing my students with this document camera, you will be
giving them the opportunity to view multiple solutions to math problems,
see other students' work, justify answers, learn effective note taking sk
ills, complete graphic organizers, learn to use a variety of math manipul
atives and view real life objects, compose classwide pieces of writing, s
ee writing modeled, edit text... and the list goes on! My students unders
tand better and are able to take content to a higher level when they have
a reference in front of them. With a document camera, I will be able to p
resent and model new material in a variety of ways to meet the needs of a
ll my studentsnannan
```

In [16]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My students come from various backgrounds in a high need urban school dis
trict Working in the Fruitvale area of Oakland we have a high percentage
of English Language Learners However all learners no matter their backgro
und can achieve and should be given the support that they need and deserv
e Our school uses Waldorf inspired teaching methods to help target instru
ction to all learners of our diverse community My students thrive when th
ey are actively engaged and taking charge of their learning When given th
e opportunity to investigate beyond books and videos I have the opportuni
ty to see my students work cooperatively think critically and let their c
reativity soar English is a second language for many of my students As re
```

```
search and experience tells us visuals are an essential part of quality t
eaching anywhere but particularly English Language Learners By providing
my students with this document camera you will be giving them the opportu
nity to view multiple solutions to math problems see other students work
justify answers learn effective note taking skills complete graphic organ
izers learn to use a variety of math manipulatives and view real life obj
ects compose classwide pieces of writing see writing modeled edit text an
d the list goes on My students understand better and are able to take con
tent to a higher level when they have a reference in front of them With a
document camera I will be able to present and model new material in a var
iety of ways to meet the needs of all my studentsnannan
```

In [17]:
```
# 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"]
```

In [18]:
```
# 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, 2055.07it/s]
```

In [19]:
```
# after preprocesing
```

```
preprocessed_essays_train[20000]
```

Out[19]:
```
'students come various backgrounds high need urban school district workin
g fruitvale area oakland high percentage english language learners howeve
r learners no matter background achieve given support need deserve school
uses waldorf inspired teaching methods help target instruction learners d
iverse community students thrive actively engaged taking charge learning
given opportunity investigate beyond books videos opportunity see student
s work cooperatively think critically let creativity soar english second
language many students research experience tells us visuals essential par
t quality teaching anywhere particularly english language learners provid
ing students document camera giving opportunity view multiple solutions m
ath problems see students work justify answers learn effective note takin
g skills complete graphic organizers learn use variety math manipulatives
view real life objects compose classwide pieces writing see writing model
ed edit text list goes students understand better able take content highe
r level reference front document camera able present model new material v
ariety ways meet needs studentsnannan'
```

Preprocessing Of Essay in Test Data

In [20]:
```
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 [00:17<00:00, 2023.00it/s]
```

In [21]:
```
preprocessed_essays_test[10000]
```

Out[21]:
```
'students walk classroom every day full excitement filled eagerness learn
kindergarteners bringing variety languages family backgrounds experiences
share classroom community enrich growth collective understandings world a
round us thinkers dreamers communicating listening questioners desire dee
per understanding world around building foundation stem realize budding s
cientists users technology engineers mathematicians future plan give grea
t start educational path school stem lab loaned classroom two robots cont
rolled students program ipads ipads covers close ipad not protect damage
dropped consequently one ipad needs replaced programming robots move ipad
mobile safe builds technology skills move generation future new mobility
learning using technology eliminate need sit still learn alone bring peer
s together learn cooperate build stem skills nannan'
```

Preprocessing Of Essay in CV Data :

In [22]:
```
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 [00:12<00:00, 1915.72it/s]
```

In [23]:
```
preprocessed_essays_cv[10000]
```

Out[23]:
```
'second grade students diverse group one thing common love learning criti
cal thinkers love immerse meaningful activities learning experiences curi
osity box thinking makes classroom interesting lively environment teach r
ural title school georgia 100 students receive free breakfast 85 receive
free reduced price lunch many families district difficulties making ends
meet funding projects present challenge second grade year great growth st
udents bridge early childhood third grade expectations increase strive ma
ke year one students learn respect work collaboratively finding joy learn
ing want help provide tools need meet full potential world becoming incre
asingly digital important prepare students future giving access technolog
y classroom students highly motivated explore create transform learning u
sing web based applications educational websites although highly encourag
e collaboration classroom times not enough resources meet needs students
two chromebooks classroom would allow engagement learning activities allo
wing reduce group size students engaged time mice would help little hands
navigate effectively nannan'
```

NUMBER OF WORDS IN EACH ESSAYS IN TRAIN , CV & TEST DATA :

In [24]:
```
noofwordsessaytrain=[]
for i in tqdm(preprocessed_essays_train):
    s = i.split(" ")
    noofwordsessaytrain.append(len(s))
print(preprocessed_essays_train[1])
noofwordsessaytrain[1]
X_train['noofwordsessay']=noofwordsessaytrain
```

```
100%|██████████| 49041/49041 [00:00<00:00, 103995.06it/s]
```

```
inner city school made hundreds bright students diverse backgrounds traum
a sensitve school classroom welcoming environment allows child feel safe
relaxed focus learning year faced room full four five year olds excited l
earn inspire teach students receive free breakfast classroom start day op
timize learning students use incubator learn life cycle hatching chicks s
pring incubation hatching classroom one amazing experiences provide stude
nts first hand view life experience students never forget hatch chicks st
udents least year one favorite things unsuccessful hatches searching bett
er equipment new incubator would make fun successful experience students
nannan
```

In [25]:
```
noofwordsessaycv=[]
for i in tqdm(preprocessed_essays_cv):
    s = i.split(" ")
    noofwordsessaycv.append(len(s))
print(preprocessed_essays_cv[1])
noofwordsessaycv[1]
X_cv['noofwordsessay']=noofwordsessaycv
```

```
100%|██████████| 24155/24155 [00:00<00:00, 95653.95it/s]
```

```
teacher high poverty low income school students face many challenges clas
sroom still expected perform level peers neighboring schools districts hi
gher incomes fewer challenges majority first graders receive free reduced
lunch cannot control goes home control experience classroom want see scho
ol loving accepting place instilling love learning early age hoping stude
nts continue grow academically throughout school career large classroom r
ug used every day several times day gathering place whole group instructi
on sharing calendar morning message stories classroom carpet top cement n
o padding right students sitting hard floor good part day also hard stude
nts stay space rug requesting individual squares marked student space sit
donating project helping create warm inviting place students learn classr
oom feel like home nannan
```

In [26]:
```
noofwordsessaytest=[]
for i in tqdm(preprocessed_essays_test):
    s = i.split(" ")
    noofwordsessaytest.append(len(s))
print(preprocessed_essays_test[1])
noofwordsessaytest[1]
X_test['noofwordsessay']=noofwordsessaytest
```

```
100%|██████████| 36052/36052 [00:00<00:00, 107751.41it/s]
```

```
students amazing individuals overcome many obstacles daily unique individ
ual attributes make special students times low confidence unable perform
academically grade level strive daily reach potential students willing tr
y anything positive attitude towards difficult tasks students anything wa
nt life give best proud students become exceed future lego therapy approp
riate approach students social communication difficulties anxiety depress
ion lego therapy shown effective approach improving social initiation pro
blem solving turn taking teamwork skills also improves work readiness fin
e motor skills lego club help students gain skills using lego inclusion s
tudents autism well children depression anxiety work together along typic
ally developing peers believe creating afterschool program lego club woul
d greatly benefit students many areas not help increase social skills bui
ld self confidence well nannan
```

1.4 Preprocessing of Project_title

In [27]:
```
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)
```

```
Put It On Our \"TAB\"let!
==================================================
When Programming and Robotics Collide
==================================================
Teaching and Learning With Pen-Pals
==================================================
ELMO, Not Just a Fuzzy Animal
==================================================
```

In [28]:
```
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 not in stopwords)
    preprocessed_projectitle_train.append(sent1.lower().strip())
```

```
100%|██████████| 49041/49041 [00:01<00:00, 46541.90it/s]
```

In [29]:
```
preprocessed_projectitle_train[5000]
```

Out[29]:
```
'we on fire for reading'
```

Preprocessing Of Project Title in CV Data :

In [30]:
```
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 not in stopwords)
    preprocessed_projectitle_cv.append(sent1.lower().strip())
```

```
100%|██████████| 24155/24155 [00:00<00:00, 43854.01it/s]
```

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

Out[31]:
```
['hokki stools active students',
 'sit see',
 'christian crew flexible seating',
 'together we learn',
 'help students succeed needed supplies']
```

Preprocessing on Project Title Test Data :

In [32]:
```
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 not in stopwords)
    preprocessed_projectitle_test.append(sent1.lower().strip())
```

```
100%|██████████| 36052/36052 [00:00<00:00, 45945.62it/s]
```

In [33]:
```
# after preprocesing
preprocessed_projectitle_test[19995:20000]
```

Out[33]:
```
['wacom we ca not illustrate digitally',
 'extra extra read all about it in science world',
 'city kids learn how their city got this way',
 'stability balls for learning success in dyslexia',
 'creating little book worms']
```

NUMBER OF WORDS IN PROJECT TITLE TRAIN, CV AND TEST DATA :

In [34]:
```
noofwordstitletrain=[]
for i in tqdm(preprocessed_projectitle_train):
    s = i.split(" ")
    noofwordstitletrain.append(len(s))
print(preprocessed_projectitle_train[1])
noofwordstitletrain[1]
X_train['noofwordstitle']=noofwordstitletrain
```

```
100%|██████████| 49041/49041 [00:00<00:00, 927789.84it/s]
```

```
egg actly what we need learn about hatching chicks
```

In [35]:
```
noofwordstitlecv=[]
for i in tqdm(preprocessed_projectitle_cv):
    s = i.split(" ")
    noofwordstitlecv.append(len(s))
print(preprocessed_projectitle_cv[1])
noofwordstitlecv[1]
X_cv['noofwordstitle']=noofwordstitlecv
```

```
100%|██████████| 24155/24155 [00:00<00:00, 1009882.31it/s]
```

```
a comfortable place learn grow
```

In [36]:
```
noofwordstitletest=[]
for i in tqdm(preprocessed_projectitle_test):
    s = i.split(" ")
    noofwordstitletest.append(len(s))
print(preprocessed_projectitle_test[1])
noofwordstitletest[1]
X_test['noofwordstitle']=noofwordstitletest
```

```
100%|██████████| 36052/36052 [00:00<00:00, 657259.43it/s]
```

```
lego club
```

DATA PREPROCESSING OF TEACHER_PREFIX IN TRAIN DATA :

In [37]:
```
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 [00:00<00:00, 52295.36it/s]
```

In [38]:
```
X_train['teacher_prefix'].fillna('',inplace=True)
```

DATA PREPROCESSING OF TEACHER_PREFIX IN CV DATA :

In [39]:
```
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, 58234.43it/s]
```

In [40]:
```
X_cv['teacher_prefix'].fillna('',inplace=True)
```

DATA PREPROCESSING OF TEACHER_PREFIX IN TEST DATA :

In [41]:
```
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, 56267.27it/s]
```

In [42]:
```
X_test['teacher_prefix'].fillna('',inplace=True)
```

1.5 Preparing data for models

In [43]:
```
project_data.columns
```

Out[43]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_stat
e',
       'project_submitted_datetime', 'project_grade_category', 'project_t
itle',
       '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. Vectorizing Categorical data

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

In [44]:
```
# 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 [45]:
```
# 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 [46]:
```
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 [47]:
```
#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 [48]:
```
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, 682949.78it/s]
100%|██████████| 24155/24155 [00:00<00:00, 692045.69it/s]
100%|██████████| 36052/36052 [00:00<00:00, 737736.18it/s]
```

In [49]:
```
set(preprocessed_pg_train)
```

Out[49]:
```
{'grades_3-5', 'grades_6-8', 'grades_9-12', 'grades_prek-2'}
```

In [50]:
```
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 [51]:
```
# We are considering only the words which appeared in at least 10 documents
(rows or projects).
vectorizerbowe = CountVectorizer(min_df=10,ngram_range=(1,2), max_features=5
000)
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, 5000)
Shape of matrix after one hot encoding  (24155, 5000)
Shape of matrix after one hot encoding  (36052, 5000)
```

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

In [52]:
```
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, 2086)
Shape of matrix after one hot encoding  (24155, 2086)
Shape of matrix after one hot encoding  (36052, 2086)
```

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

In [53]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizertie = TfidfVectorizer(min_df=10,ngram_range=(1,2), max_features=50
00)
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, 5000)
Shape of matrix after one hot encoding  (24155, 5000)
Shape of matrix after one hot encoding  (36052, 5000)
```

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

In [54]:
```
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, 2086)
Shape of matrix after one hot encoding  (24155, 2086)
Shape of matrix after one hot encoding  (36052, 2086)
```

Using Pretrained Models: Avg W2V

In [55]:
```
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084
039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')
words = []
for i in preprocessed_essays_train:
    words.extend(i.split(' '))
for i in preprocessed_projectitle_train:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))
inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our co
upus", \
```

```
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")
words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))
# stronging variables into pickle files python: http://www.jessicayung.com/h
ow-to-use-pickle-to-save-and-load-variables-in-python/
import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)
```

```
Loading Glove Model
```

```
1917495it [03:57, 8079.80it/s]
```

```
Done. 1917495  words loaded!
all the words in the coupus 6997295
the unique words in the coupus 43232
The number of words that are present in both glove vectors and our coupus
39425 ( 91.194 %)
word 2 vec length 39425
```

In [56]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/h
ow-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

AVG W2V on ESSAY IN TRAIN , CV & TEST DATA :

In [57]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_essay_train = []; # the avg-w2v for each sentence/review is
 stored in this list
for sentence in tqdm(preprocessed_essays_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_essay_train.append(vector)

print(len(avg_w2v_vectors_essay_train))
print(len(avg_w2v_vectors_essay_train[0]))
```

```
100%|██████████| 49041/49041 [00:12<00:00, 3849.52it/s]
```

```
49041
300
```

In [58]:
```
avg_w2v_vectors_essay_cv = []; # the avg-w2v for each sentence/review is sto
red in this list
```

```
for sentence in tqdm(preprocessed_essays_cv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_essay_cv.append(vector)

print(len(avg_w2v_vectors_essay_cv))
print(len(avg_w2v_vectors_essay_cv[0]))
```

```
100%|██████████| 24155/24155 [00:06<00:00, 3868.06it/s]
```

```
24155
300
```

In [59]:
```
avg_w2v_vectors_essay_test = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm(preprocessed_essays_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_essay_test.append(vector)

print(len(avg_w2v_vectors_essay_test))
print(len(avg_w2v_vectors_essay_test[0]))
```

```
100%|██████████| 36052/36052 [00:08<00:00, 4229.97it/s]
```

```
36052
300
```

AVG W2V on Project Title in Train , Test & CV Data :

In [60]:
```
avg_w2v_vectors_title_train = []; # the avg-w2v for each sentence/review is
 stored in this list
for sentence in tqdm(preprocessed_projectitle_train): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_train.append(vector)

print(len(avg_w2v_vectors_title_train))
print(len(avg_w2v_vectors_title_train[0]))
```

```
100%|██████████| 49041/49041 [00:00<00:00, 75921.23it/s]
```

```
49041
300
```

In [61]:
```
avg_w2v_vectors_title_cv = []; # the avg-w2v for each sentence/review is sto
red in this list
for sentence in tqdm(preprocessed_projectitle_cv): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_cv.append(vector)

print(len(avg_w2v_vectors_title_cv))
print(len(avg_w2v_vectors_title_cv[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 62443.63it/s]
```

```
24155
300
```

In [62]:
```
avg_w2v_vectors_title_test = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm(preprocessed_projectitle_test): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the essay
    for word in sentence.split(): # for each word in a essay
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_title_test.append(vector)

print(len(avg_w2v_vectors_title_test))
print(len(avg_w2v_vectors_title_test[0]))
```

```
100%|██████████| 36052/36052 [00:00<00:00, 67929.69it/s]
```

```
36052
300
```

Using Pretrained Models: TFIDF weighted W2V on ESSAY in TRAIN , CV & TEST Data :

In [63]:
```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays_train)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [64]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_essay_train = []; # the avg-w2v for each sentence/review i
s stored in this list
for sentence in tqdm(preprocessed_essays_train): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
```

```
    tf_idf_weight =0; # num of words with a valid vector in the sentence/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_essay_train.append(vector)

print(len(tfidf_w2v_vectors_essay_train))
print(len(tfidf_w2v_vectors_essay_train[0]))
```

```
100%|██████████| 49041/49041 [01:18<00:00, 623.29it/s]
```

```
49041
300
```

In [65]:
```
tfidf_w2v_vectors_essay_cv = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm(preprocessed_essays_cv): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_essay_cv.append(vector)

print(len(tfidf_w2v_vectors_essay_cv))
print(len(tfidf_w2v_vectors_essay_cv[0]))
```

```
100%|██████████| 24155/24155 [00:39<00:00, 618.21it/s]
```

```
24155
300
```

In [66]:
```
tfidf_w2v_vectors_essay_test = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm(preprocessed_essays_test): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
```

```
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_essay_test.append(vector)

print(len(tfidf_w2v_vectors_essay_test))
print(len(tfidf_w2v_vectors_essay_test[0]))
```

```
100%|██████████| 36052/36052 [00:59<00:00, 607.34it/s]
```

```
36052
300
```

TFIDF weighted W2V on Project_Title in TRAIN , TEST & CV Data :

In [67]:
```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_projectitle_train)
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [68]:
```
tfidf_w2v_vectors_title_train = []; # the avg-w2v for each sentence/review i
s stored in this list
for sentence in tqdm(preprocessed_projectitle_train): # for each review/sent
ence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_train.append(vector)

print(len(tfidf_w2v_vectors_title_train))
print(len(tfidf_w2v_vectors_title_train[0]))
```

```
100%|██████████| 49041/49041 [00:01<00:00, 37757.22it/s]
```

```
49041
300
```

In [69]:
```
tfidf_w2v_vectors_title_cv = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm(preprocessed_projectitle_cv): # for each review/sentenc
e
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
```

```
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_cv.append(vector)

print(len(tfidf_w2v_vectors_title_cv))
print(len(tfidf_w2v_vectors_title_cv[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 27805.47it/s]
```

```
24155
300
```

In [70]:
```
tfidf_w2v_vectors_title_test = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm(preprocessed_projectitle_test): # for each review/sente
nce
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/rev
iew
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spl
it())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_test.append(vector)

print(len(tfidf_w2v_vectors_title_test))
print(len(tfidf_w2v_vectors_title_test[0]))
```

```
100%|██████████| 36052/36052 [00:00<00:00, 36354.77it/s]
```

```
36052
300
```

Vectorizing Numerical features

1) PRICE

In [71]:
```
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
price_scalar = StandardScaler()
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 [72]:
```
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 [73]:
```
quantity_scaler = StandardScaler()
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 StandardScaler.

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

Data with input dtype int64 was converted to float64 by StandardScaler.

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

Data with input dtype int64 was converted to float64 by StandardScaler.

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

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

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

In [74]:
```
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 [75]:
```
noofprojects = StandardScaler()
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 StandardScaler.

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

Data with input dtype int64 was converted to float64 by StandardScaler.

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

Data with input dtype int64 was converted to float64 by StandardScaler.

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

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

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

4) Number Of Words In ESSAY :

In [76]:
```
noofwordse = StandardScaler()
noofwordse.fit(X_train['noofwordsessay'].values.reshape(-1,1))
noessay_train = noofwordse.transform(X_train['noofwordsessay'].values.reshap
e(-1,1))
noessay_cv = noofwordse.transform(X_cv['noofwordsessay'].values.reshape(-1,1
))
noessay_test = noofwordse.transform(X_test['noofwordsessay'].values.reshape(
-1,1))
print(noessay_train.shape, y_train.shape)
print(noessay_cv.shape, y_cv.shape)
print(noessay_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 StandardScaler.

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

Data with input dtype int64 was converted to float64 by StandardScaler.

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

Data with input dtype int64 was converted to float64 by StandardScaler.

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

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

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

5) Number Of Words In TITLE :

In [77]:
```
noofwordst = StandardScaler()
noofwordst.fit(X_train['noofwordstitle'].values.reshape(-1,1))
notitle_train = noofwordst.transform(X_train['noofwordstitle'].values.reshap
e(-1,1))
notitle_cv = noofwordst.transform(X_cv['noofwordstitle'].values.reshape(-1,1
))
notitle_test = noofwordst.transform(X_test['noofwordstitle'].values.reshape(
-1,1))
print(notitle_train.shape, y_train.shape)
print(notitle_cv.shape, y_cv.shape)
print(notitle_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 StandardScaler.

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

Data with input dtype int64 was converted to float64 by StandardScaler.

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

Data with input dtype int64 was converted to float64 by StandardScaler.

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

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

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

```
(36052, 1) (36052,)
```

Merging all the above features

SET 1 :

In [78]:
```
from scipy.sparse import hstack
X_tr = hstack((text_bow_train,title_bow_train,school_state_categories_one_hot_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,pgc_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, 7188) (49041,)
(24155, 7188) (24155,)
(36052, 7188) (36052,)
```

SET 2 :

In [79]:
```
X_tr2 = hstack((title_tfidf_train,text_tfidf_train,school_state_categories_one_hot_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_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_cv,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,quantity_test)).tocsr()

print("Final Data matrix")
print(X_tr2.shape, y_train.shape)
print(X_cr2.shape, y_cv.shape)
print(X_te2.shape, y_test.shape)
```

```
Final Data matrix
(49041, 7198) (49041,)
(24155, 7198) (24155,)
(36052, 7198) (36052,)
```

SET 3 :

In [79]:
```
X_tr3 = hstack((avg_w2v_vectors_essay_train,avg_w2v_vectors_title_train,scho
ol_state_categories_one_hot_train,pgc_one_hot_train,teacher_prefix_categorie
s_one_hot_train,categories_one_hot_train,sub_categories_one_hot_train,prev_p
rojects_train,price_train,quantity_train)).tocsr()
X_cr3 = hstack((avg_w2v_vectors_essay_cv,avg_w2v_vectors_title_cv,school_sta
te_categories_one_hot_cv,pgc_one_hot_cv,teacher_prefix_categories_one_hot_cv
,categories_one_hot_cv,sub_categories_one_hot_cv,prev_projects_cv,price_cv,q
uantity_cv)).tocsr()
X_te3 = hstack((avg_w2v_vectors_essay_test,avg_w2v_vectors_title_test,school
_state_categories_one_hot_test,pgc_one_hot_test,teacher_prefix_categories_on
e_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_tr3.shape, y_train.shape)
print(X_cr3.shape, y_cv.shape)
print(X_te3.shape, y_test.shape)
```

```
Final Data matrix
(49041, 702) (49041,)
(24155, 702) (24155,)
(36052, 702) (36052,)
```

SET 4 :

In [80]:
```
X_tr4 = hstack((tfidf_w2v_vectors_essay_train,tfidf_w2v_vectors_title_train,
school_state_categories_one_hot_train,pgc_one_hot_train,teacher_prefix_categ
ories_one_hot_train,categories_one_hot_train,sub_categories_one_hot_train,pr
ev_projects_train,price_train,quantity_train)).tocsr()
X_cr4 = hstack((tfidf_w2v_vectors_essay_cv,tfidf_w2v_vectors_title_cv,school
_state_categories_one_hot_cv,pgc_one_hot_cv,teacher_prefix_categories_one_ho
t_cv,categories_one_hot_cv,sub_categories_one_hot_cv,prev_projects_cv,price_
cv,quantity_cv)).tocsr()
X_te4 = hstack((tfidf_w2v_vectors_essay_test,tfidf_w2v_vectors_title_test,sc
hool_state_categories_one_hot_test,pgc_one_hot_test,teacher_prefix_categorie
s_one_hot_test,categories_one_hot_test,sub_categories_one_hot_test,prev_proj
ects_test,price_test,quantity_test)).tocsr()

print("Final Data matrix")
print(X_tr4.shape, y_train.shape)
print(X_cr4.shape, y_cv.shape)
print(X_te4.shape, y_test.shape)
```

```
Final Data matrix
(49041, 702) (49041,)
(24155, 702) (24155,)
(36052, 702) (36052,)
```

Computing Sentiment Scores :

In [81]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
essay_neg_train=[]
essay_pos_train=[]
essay_neu_train=[]
essay_com_train=[]
sid = SentimentIntensityAnalyzer()
```

```
for i in preprocessed_essays_train:
    for_sentiment = i
    ss=sid.polarity_scores(for_sentiment)
    essay_neg_train.append(ss['neg'])
    essay_pos_train.append(ss['pos'])
    essay_neu_train.append(ss['neu'])
    essay_com_train.append(ss['compound'])
len(essay_neg_train)
X_train['essay_neg_train']=essay_neg_train
X_train['essay_pos_train']=essay_pos_train
X_train['essay_neu_train']=essay_neu_train
X_train['essay_com_train']=essay_com_train
```

```
F:\Anaconda3\lib\site-packages\nltk\twitter\__init__.py:20: UserWarning:

The twython library has not been installed. Some functionality from the t
witter package will not be available.
```

```
[nltk_data] Downloading package vader_lexicon to C:\Users\Mitadru
[nltk_data]     Ghosh\AppData\Roaming\nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

In [82]:
```
essay_neg_cv=[]
essay_pos_cv=[]
essay_neu_cv=[]
essay_com_cv=[]
sid = SentimentIntensityAnalyzer()
for i in preprocessed_essays_cv:
    for_sentiment = i
    ss=sid.polarity_scores(for_sentiment)
    essay_neg_cv.append(ss['neg'])
    essay_pos_cv.append(ss['pos'])
    essay_neu_cv.append(ss['neu'])
    essay_com_cv.append(ss['compound'])
len(essay_neg_cv)
X_cv['essay_neg_cv']=essay_neg_cv
X_cv['essay_pos_cv']=essay_pos_cv
X_cv['essay_neu_cv']=essay_neu_cv
X_cv['essay_com_cv']=essay_com_cv
```

In [83]:
```
essay_neg_test=[]
essay_pos_test=[]
essay_neu_test=[]
essay_com_test=[]
sid = SentimentIntensityAnalyzer()
for i in preprocessed_essays_test:
    for_sentiment = i
    ss=sid.polarity_scores(for_sentiment)
    essay_neg_test.append(ss['neg'])
    essay_pos_test.append(ss['pos'])
    essay_neu_test.append(ss['neu'])
    essay_com_test.append(ss['compound'])
len(essay_neg_test)
X_test['essay_neg_test']=essay_neg_test
X_test['essay_pos_test']=essay_pos_test
X_test['essay_neu_test']=essay_neu_test
X_test['essay_com_test']=essay_com_test
```

In [84]:
```
essay_neg_test=X_test['essay_neg_test'].values.reshape(-1,1)
essay_pos_test=X_test['essay_pos_test'].values.reshape(-1,1)
```

```
essay_neu_test=X_test['essay_pos_test'].values.reshape(-1,1)
essay_com_test=X_test['essay_pos_test'].values.reshape(-1,1)
essay_neg_cv=X_cv['essay_pos_cv'].values.reshape(-1,1)
essay_pos_cv=X_cv['essay_pos_cv'].values.reshape(-1,1)
essay_neu_cv=X_cv['essay_pos_cv'].values.reshape(-1,1)
essay_com_cv=X_cv['essay_pos_cv'].values.reshape(-1,1)
essay_neg_train=X_train['essay_neg_train'].values.reshape(-1,1)
essay_pos_train=X_train['essay_pos_train'].values.reshape(-1,1)
essay_neu_train=X_train['essay_neu_train'].values.reshape(-1,1)
essay_com_train=X_train['essay_com_train'].values.reshape(-1,1)
```

SET 5 :

In [85]:
```
from scipy.sparse import hstack
X_tr5 = hstack((essay_neg_train,essay_pos_train,essay_neu_train,essay_com_train,noessay_train,notitle_train,school_state_categories_one_hot_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_cr5 = hstack((essay_neg_cv,essay_pos_cv,essay_neu_cv,essay_com_cv,noessay_cv,notitle_cv,school_state_categories_one_hot_cv,pgc_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_te5 = hstack((essay_neg_test,essay_pos_test,essay_neu_test,essay_com_test,noessay_test,notitle_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_tr5.shape, y_train.shape)
print(X_cr5.shape, y_cv.shape)
print(X_te5.shape, y_test.shape)
```

```
Final Data matrix
(49041, 108) (49041,)
(24155, 108) (24155,)
(36052, 108) (36052,)
```

Assignment 5: Logistic Regression

1. **[Task-1] Logistic Regression(either SGDClassifier with log loss, or LogisticRegression) on these feature sets**
   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (`BOW with bi-grams` with `min_df=10` and `max_features=5000`)
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (`TFIDF with bi-grams` with `min_df=10` and `max_features=5000`)
   - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
   - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)
2. **Hyper paramter tuning (find best hyper parameters corresponding the algorithm that you choose)**
   - Find the best hyper parameter which will give the maximum AUC value

- 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. **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.

- 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.

4. **[Task-2] Apply Logistic Regression on the below feature set Set 5 by finding the best hyper parameter as suggested in step 2 and step 3.**
5. Consider these set of features Set 5 :

- **school_state** : categorical data
- **clean_categories** : categorical data
- **clean_subcategories** : categorical data
- **project_grade_category** :categorical data
- **teacher_prefix** : categorical data
- **quantity** : numerical data
- **teacher_number_of_previously_posted_projects** : numerical data
- **price** : numerical data
- **sentiment score's of each of the essay** : numerical data
- **number of words in the title** : numerical data
- **number of words in the combine essays** : numerical data

And apply the Logistic regression on these features by finding the best hyper paramter as suggested in step 2 and step 3

6. **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

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakage, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link.

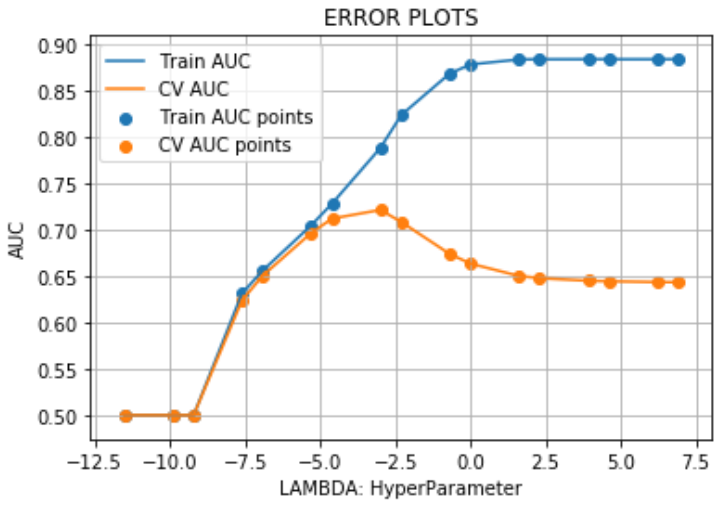
2. Logistic Regression

Logistic Regression On SET 1 :

In [87]:
```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

train_auc = []
cv_auc = []
log_lambdas =[]

lambdas = [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, 500,1000]

for i in tqdm(lambdas):
    lr = LogisticRegression(C=i,penalty='l1',solver='liblinear',class_weight
= 'balanced')
    lr.fit(X_tr, y_train)
    y_train_pred = lr.predict_proba(X_tr)[:,1]
    y_cv_pred = lr.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))
```

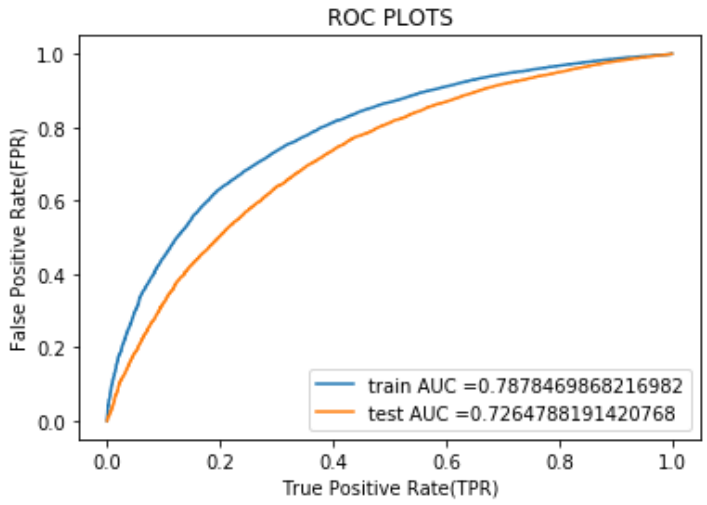
```
100%|██████████| 17/17 [10:49<00:00, 67.23s/it]
```

In [88]:
```
import numpy
plt.plot(numpy.log(lambdas), train_auc, label='Train AUC')
plt.plot(numpy.log(lambdas), cv_auc, label='CV AUC')

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

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



In [89]:
```
best_inverselambda = 0.05
```

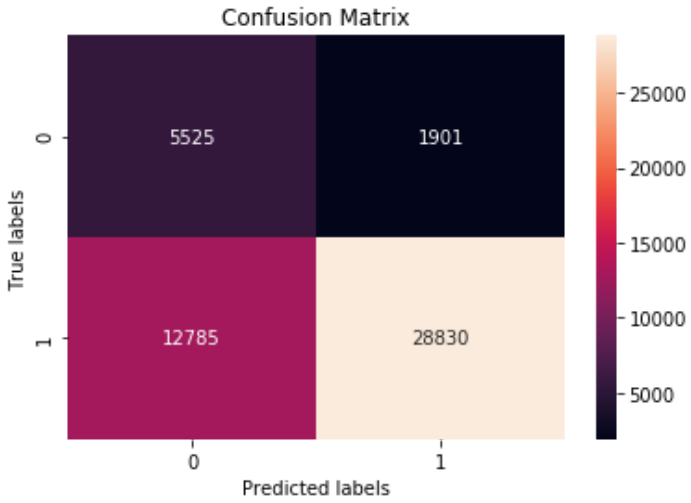
In [90]:
```
from sklearn.metrics import roc_curve, auc
ne = LogisticRegression(C=best_inverselambda,penalty='l1',solver='liblinear'
,class_weight = 'balanced')
ne.fit(X_tr,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates of the positive class
# not the predicted outputs
train_fpr, train_tpr, thresholds = roc_curve(y_train, ne.predict_proba(X_tr)
[:,1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, ne.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)
```



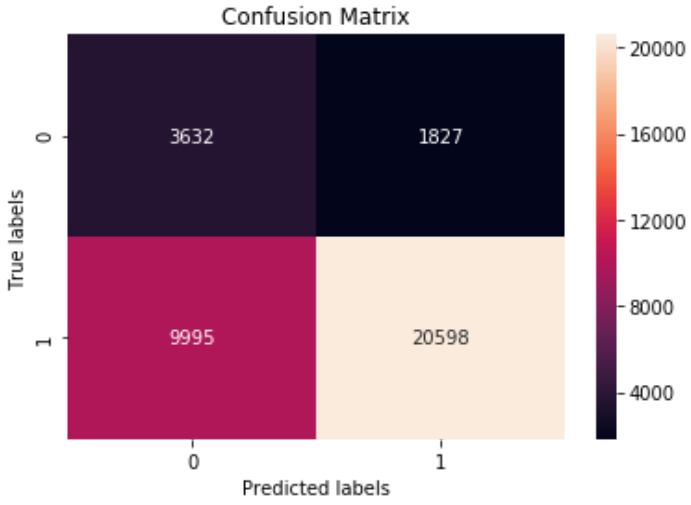

```
====================================================================================
============================
```

CONFUSION MATRIX :

In [91]:
```
#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, ne.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 [92]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, ne.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');
```



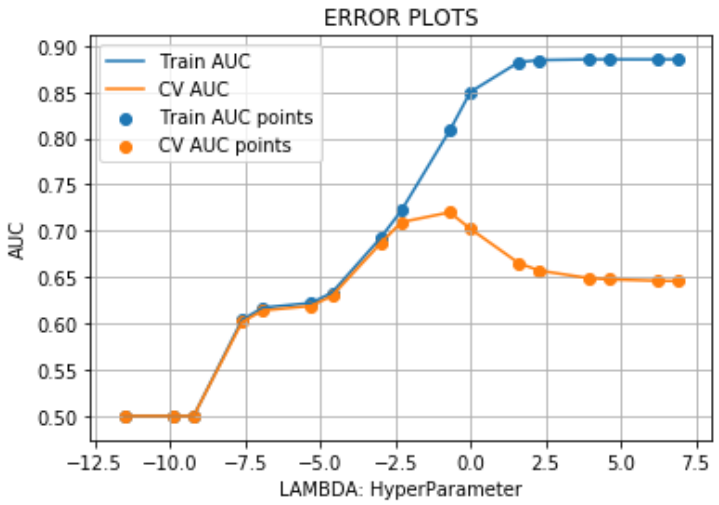
Logistic Regression On SET 2 :

In [93]:
```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

train_auc = []
cv_auc = []
log_lambdas =[]

lambdas = [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, 500,1000]
```

```
for i in tqdm(lambdas):
    lr = LogisticRegression(C=i,penalty='l1',solver='liblinear',class_weight
= 'balanced')
    lr.fit(X_tr2, y_train)
    y_train_pred = lr.predict_proba(X_tr2)[:,1]
    y_cv_pred = lr.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%|██████████| 17/17 [26:20<00:00, 150.44s/it]
```

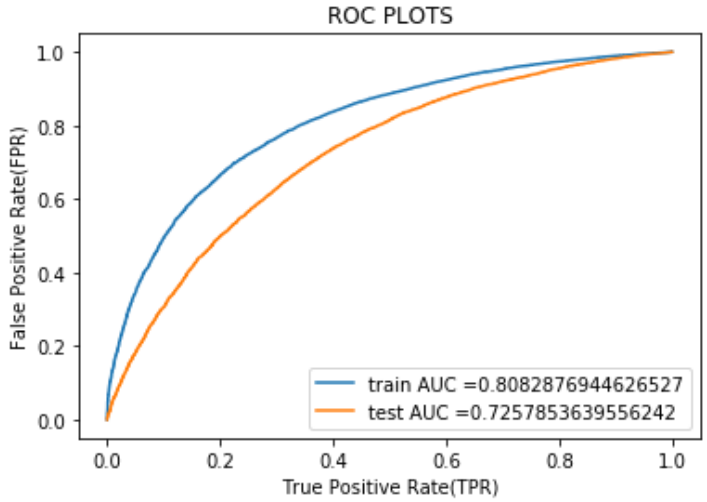
In [94]:
```
import numpy
plt.plot(numpy.log(lambdas), train_auc, label='Train AUC')
plt.plot(numpy.log(lambdas), cv_auc, label='CV AUC')

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

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



In [96]:
```
best_inverselambda = 0.5
```

In [97]:
```
from sklearn.metrics import roc_curve, auc
ne = LogisticRegression(C=best_inverselambda,penalty='l1',solver='liblinear'
,class_weight = 'balanced')
ne.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, ne.predict_proba(X_tr2
)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, ne.predict_proba(X_te2)
[:,1])
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_
tpr)))
```

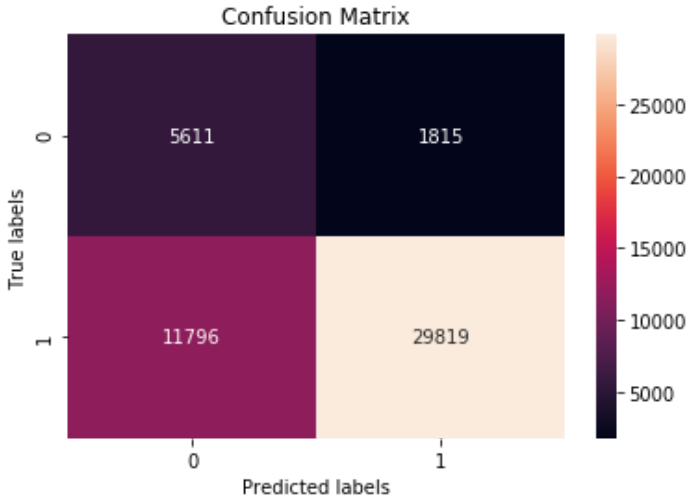
```
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr
)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("ROC PLOTS")
plt.show()
print("="*100)
```



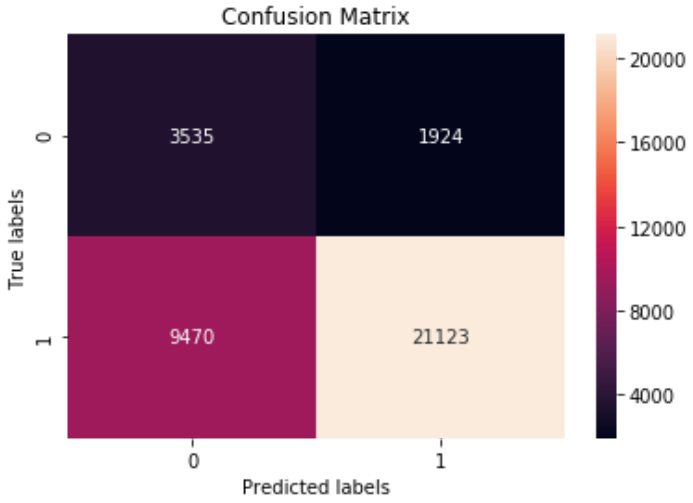
==========

CONFUSION MATRIX :

In [98]:
```
#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, ne.predict(X_tr2)), annot=True, ax = a
x,fmt='g');

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



In [99]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, ne.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');
```



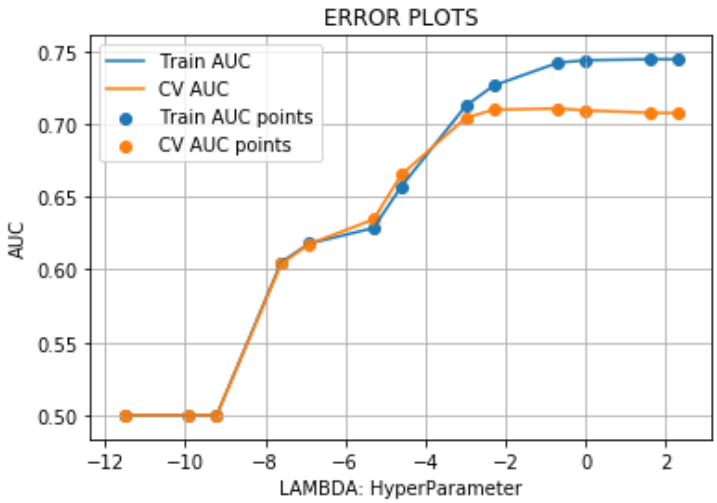
Logistic Regression On SET 3 :

In [86]:
```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

train_auc = []
cv_auc = []
log_lambdas =[]

lambdas = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10]

for i in tqdm(lambdas):
    lr = LogisticRegression(C=i,penalty='l1',solver='liblinear',class_weight = 'balanced')
    lr.fit(X_tr3, y_train)
    y_train_pred = lr.predict_proba(X_tr3)[:,1]
    y_cv_pred = lr.predict_proba(X_cr3)[:,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%|██████████| 13/13 [4:16:28<00:00, 3037.15s/it]
```

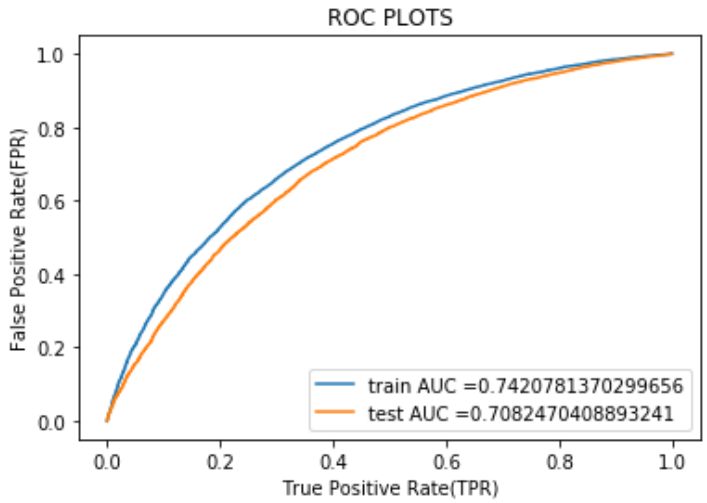
In [87]:
```
import numpy
plt.plot(numpy.log(lambdas), train_auc, label='Train AUC')
```

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

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

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



In [88]:
```
best_inverselambda = 0.5
```

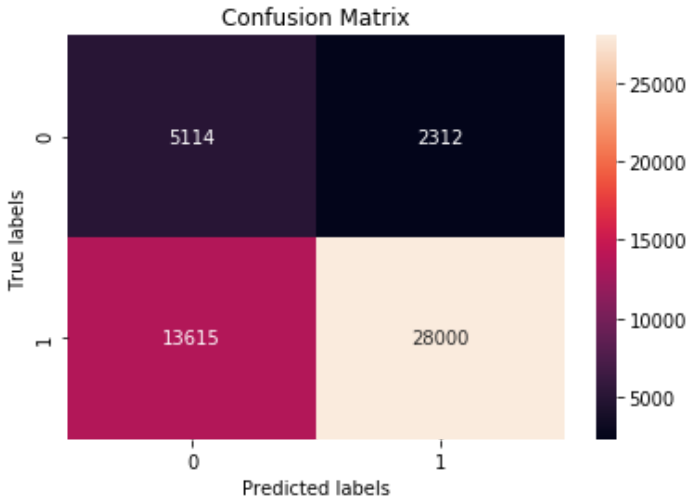
In [89]:
```
from sklearn.metrics import roc_curve, auc
ne = LogisticRegression(C=best_inverselambda,penalty='l1',solver='liblinear'
,class_weight = 'balanced')
ne.fit(X_tr3,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates of the positive class
# not the predicted outputs
train_fpr, train_tpr, thresholds = roc_curve(y_train, ne.predict_proba(X_tr3
)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, ne.predict_proba(X_te3)
[:,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)
```



==========

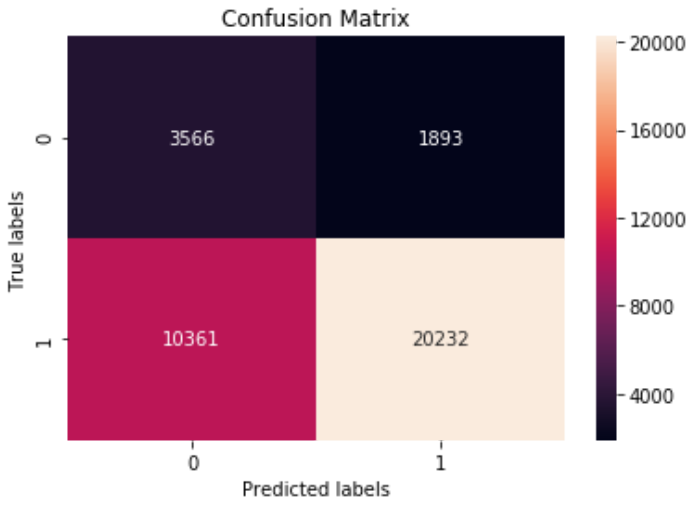
CONFUSION MATRIX :

In [90]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_train, ne.predict(X_tr3)), 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 [91]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, ne.predict(X_te3)), 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');
```



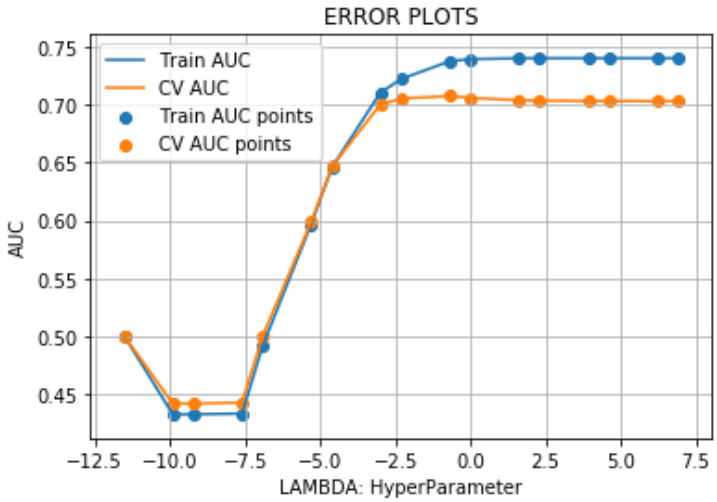
Logistic Regression On SET 4 :

In [155]:
```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

train_auc = []
cv_auc = []
log_lambdas =[]

lambdas = [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, 500,1000]

for i in tqdm(lambdas):
    lr = LogisticRegression(C=i,penalty='l1',solver='liblinear',class_weight
= 'balanced')
    lr.fit(X_tr4, y_train)
    y_train_pred = lr.predict_proba(X_tr4)[:,1]
    y_cv_pred = lr.predict_proba(X_cr4)[:,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%|██████████| 17/17 [17:10<00:00, 104.04s/it]
```

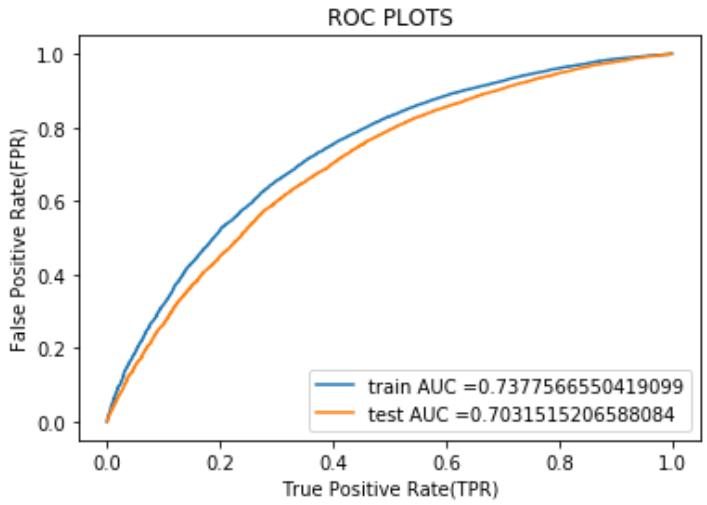
In [156]:
```
import numpy
plt.plot(numpy.log(lambdas), train_auc, label='Train AUC')
plt.plot(numpy.log(lambdas), cv_auc, label='CV AUC')

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

plt.legend()
plt.xlabel("LAMBDA: HyperParameter")
plt.ylabel("AUC")
```

```
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [157]:
```
best_inverselambda = 0.5
```

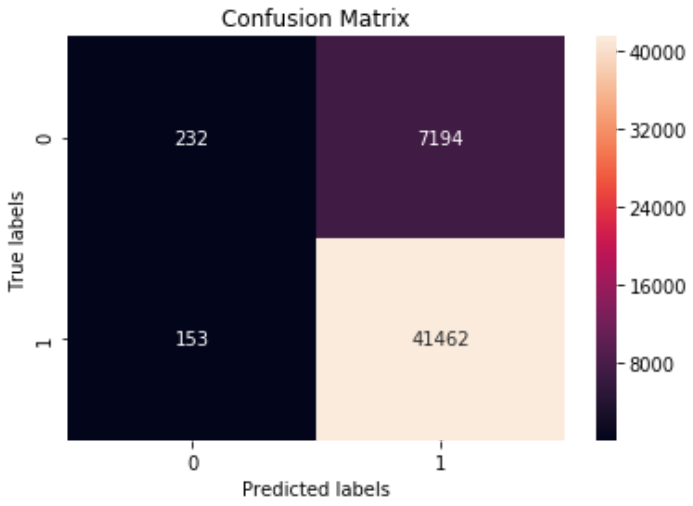
In [158]:
```
from sklearn.metrics import roc_curve, auc
ne = LogisticRegression(C=best_inverselambda,penalty='l1',solver='liblinear'
,class_weight = 'balanced')
ne.fit(X_tr4,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates of the positive class
# not the predicted outputs
train_fpr, train_tpr, thresholds = roc_curve(y_train, ne.predict_proba(X_tr4
)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, ne.predict_proba(X_te4)
[:,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)
```



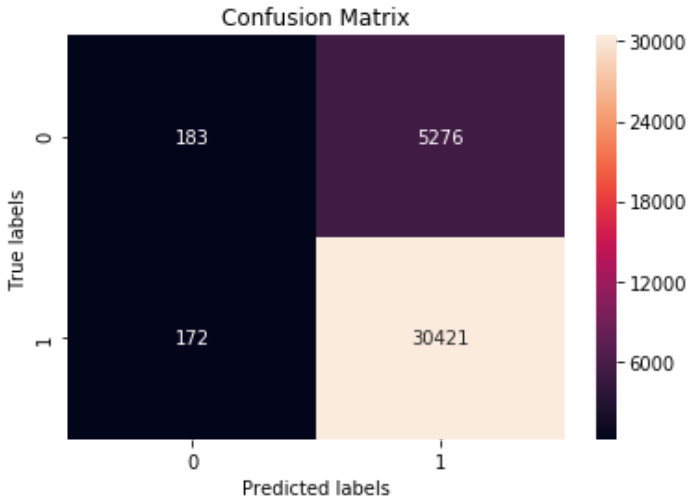
==========

CONFUSION MATRIX :

In [160]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_train, ne.predict(X_tr4)), 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 [161]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, ne.predict(X_te4)), 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');
```

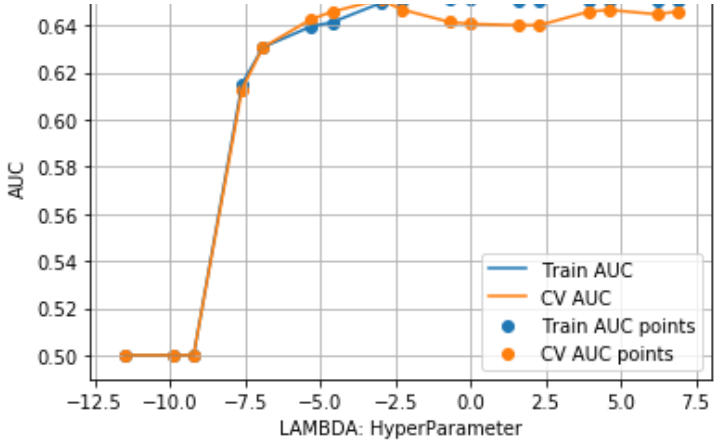


Logistic Regression On SET 5 :

In [92]:
```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score

train_auc = []
cv_auc = []
log_lambdas =[]

lambdas = [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, 500,1000]

for i in tqdm(lambdas):
    lr = LogisticRegression(C=i,penalty='l1',solver='liblinear',class_weight
= 'balanced')
    lr.fit(X_tr5, y_train)
    y_train_pred = lr.predict_proba(X_tr5)[:,1]
    y_cv_pred = lr.predict_proba(X_cr5)[:,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))
```

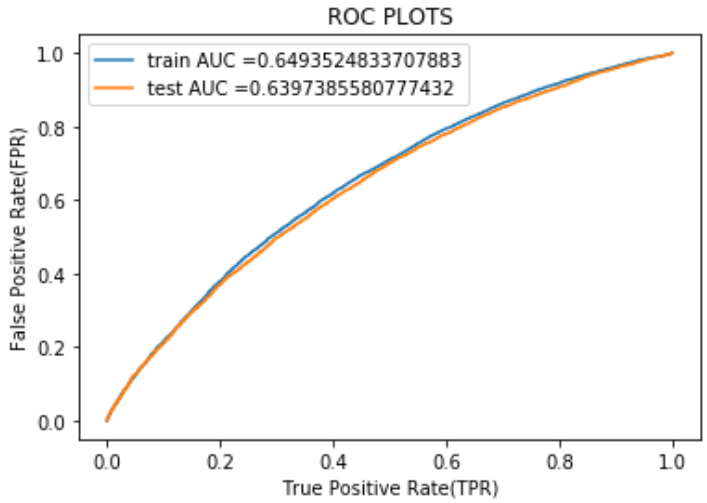
```
100%|██████████| 17/17 [10:09<00:00, 121.75s/it]
```

In [93]:
```
import numpy
plt.plot(numpy.log(lambdas), train_auc, label='Train AUC')
plt.plot(numpy.log(lambdas), cv_auc, label='CV AUC')

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

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



In [94]:
```
best_inverselambda = 0.05
```

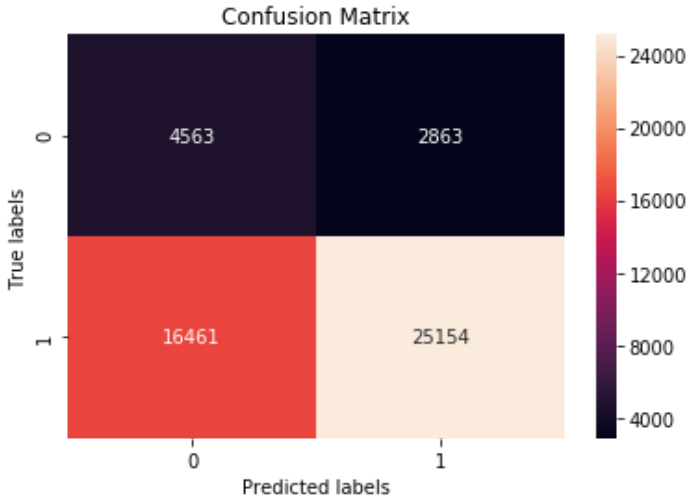
In [95]:
```
from sklearn.metrics import roc_curve, auc
ne = LogisticRegression(C=best_inverselambda,penalty='l1',solver='liblinear'
,class_weight = 'balanced')
ne.fit(X_tr5,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
imates of the positive class
# not the predicted outputs
train_fpr, train_tpr, thresholds = roc_curve(y_train, ne.predict_proba(X_tr5
)[:,1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, ne.predict_proba(X_te5)
[:,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)
```



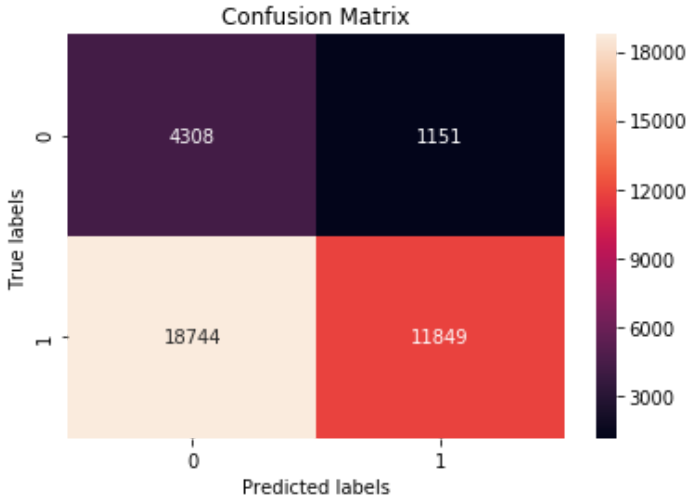
==========

CONFUSION MATRIX :

In [96]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_train, ne.predict(X_tr5)), 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 [97]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, ne.predict(X_te5)), 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');
```



Conclusion :

In [98]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Hyper Parameter", "AUC"]

x.add_row(["BOW(USING STANDARD SCALER)",0.05,72.6])
x.add_row(["TFIDF(USING STANDARD SCALER)",0.5,72.3])
x.add_row(["AVG W2V(USING STANDARD SCALER)",0.5,70.9])
x.add_row(["TFIDF W2V(USING STANDARD SCALER)",0.5,70])
x.add_row(["NO OF WORDS/SENTIMENT SCORES",0.05,63.5])
print(x.get_string(titles = "Logistic Regression - Observations"))
```

```
+-----------------------------------+-----------------+------+
|            Vectorizer             | Hyper Parameter | AUC  |
+-----------------------------------+-----------------+------+
|    BOW(USING STANDARD SCALER)     |       0.05      | 72.6 |
|   TFIDF(USING STANDARD SCALER)    |       0.5       | 72.3 |
|  AVG W2V(USING STANDARD SCALER)   |       0.5       | 70.9 |
| TFIDF W2V(USING STANDARD SCALER)  |       0.5       |  70  |
|   NO OF WORDS/SENTIMENT SCORES    |       0.05      | 63.5 |
+-----------------------------------+-----------------+------+
```