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

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]))
```

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
]))
```

Text Preprocessing

ESSAY

In [7]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

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

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	p
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	51466	p167075	97fcea2e33bdb2bb51407d799f425d8e	Ms.	CA	2
1	144977	p213678	60d683391ede8829c3f23b90e9f2bcf2	Ms.	MA	2
2	167723	p051648	2de954340b83289b322a80f2cd641cb8	Ms.	IL	2

3	125271	p204888	02393bb54a1e4a3ec19c720a72f6bd51	Ms.	NY	2
4	181832	p176949	d0bc361cbd021a4347900d18e1d59e17	Ms.	MA	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 classroom is full of joy and creativity! For many of the students that
take my class, art is not something they have allot of experience with, h
owever once they see the results of their first project, they are hooked!
Students continuously create,evaluate and reflect on their own work!\r\n
\r\nMy students are from very diverse backgrounds, they range in age from
14 to 19 and classes are mixes of 9th, 10th, 11th and 12th graders.\r\nMa
ny are from socio-economically distressed backgrounds, most are on the fr
ee-lunch program and quite a few have jobs that begin in the am before sc
hool or start right after the finish of the school day.  We've set the ba
r high in terms of expectations of work, and by supplying the art room, t
his year's students will continue to produce quality artwork, which incre
ases their confidence level and pride in their work. Even with these extr
a burdens that many in the surrounding areas do not face, my students are
all heart, they are just great young people! It's an honor and a pleasure
each day to engage and interact with such determined kids, their unique p
erspectives on each project inspires!\r\n\r\n\r\n\r\nFor the past four ye
ars I've awarded students who love to draw with a quality sketchbook, not
hing describes how happy they are to receive this beautiful book! Many wo
rk on lined paper and have a collection of loved sketches collected in th
e bottom of their back pack. the sketch book allows them a special place
to create and express themselves. Erasers and paper are the building bloc
ks of an art class, we are always  in need of them and having them readil
y at hand allows students to move through an assignment with out worry. Q
uality brushes make all the difference to my inspiring artists, nothing i
s worse than trying to paint with a thread-bear brush, these brushes will
be used to investigate artistic movements, and painting techniques throug
h out history. The canvases will allow us as a class in advanced art to c
reate works that are worthy of display. Art is a subject that builds mora
l and confidence, when a child receives materials that are are of value t
hey treat their work equally!nannan
==================================================
My students are awesome! For the 2015-2016 school year, I had the opportu
nity to watch them grow from below grade level second graders to many of
```

```
them above grade level second graders. For the 2016-2017 school year I am
able to have an opportunity most teachers don't get....I get to keep my c
lass and move with them to third grade. \r\n\r\nMany of my students come
from poverty. We offer 100% free breakfast, lunch and snacks. We also hav
e a diverse population of students representing many countries and cultur
es around the world.We need higher level books for our library for next y
ear. As a second grade teacher, I don't have many books that are third an
d fourth grade reading levels. Since I am moving with my students to thir
d grade, I want to keep up the momentum and success they have had this ye
ar by having more higher level texts for them. This past year the majorit
y of my students grew their reading levels by at least 2 years. I strongl
y feel with the success we have had, they will easily grow another two ye
ars in third grade if we have the resources.nannan
==================================================
You never know what your day is going to be like when you enter the doors
of our school. Majority of our students come from challenging backgrounds
and come to school seeking stability, love, safety, and comfort. We are t
he largest elementary school in our district. We are a Title 1 school con
taining 94% economically disadvantaged students, 20% ESL, and 13% in spec
ial education. Our school goes above and beyond to help our students achi
eve success and give them the skills they need to reach their goals. We w
ant them to feel at home in our classrooms to help them better focus on a
cademics.The wobble stools and exercise disks allow students to get those
wiggles out, while still remaining focused on the task in front of them.
Lap desks are for students who prefer to learn on the floor, but still ne
ed a hard surface to write on. The bean bag chairs are another fun seatin
g option where students can sit down with a good book or complete assignm
ents. \r\n\r\nThese seating options will inspire collaborative groups in
the classroom. Students learn better when they are working in an area tha
t they feel comfortable. We want to create an environment where students
have multiple seating options, so they can choose where they learn their
best. Instead of students sitting at a traditional desk and chair, flexib
le seating offers students a more engaging way to learn challenging conce
pts. \r\n\r\nThe purpose of flexible seating is to increase student focus
and allowing the students to be more in charge of their own learning. By
choosing their own seat, students are in charge of deciding where they ca
n learn successfully.nannan
==================================================
\r\nMany of our students come from single parent and low income homes wit
h limited transportation. We have a large ESL and special education popul
ation that is growing each year. In addition, many of these students are
new immigrants with very limited English language skills.\r\n\r\n     The
majority of our students qualify for free and reduced lunch and are from
85% minority populations. When looking at different indicators, almost al
l of our students are at risk and in need of intervention and support.\r
\n\r\nThis project is for our non English speaking students. Many are str
uggling in their coursework due to the language barrier. With these dicti
onaries, our students will be able to better complete assignments they do
not understand. They will be able to complete work independently by looki
ng up the equivalent words in Spanish for a better understanding of the m
aterial. \r\n\r\nThese dictionaries are also approved for state standardi
zed testing; therefore, students will not only be able to use them in the
classroom but also for testing. The impact projected is better performanc
e in the classroom, testing, and learning the English language.nannan
==================================================
```

In [13]:

```
# 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]:
```
sent = decontracted(X_train['essay'].values[20000])
print(sent)
print("="*50)
```

```
\r\nMany of our students come from single parent and low income homes wit
h limited transportation. We have a large ESL and special education popul
ation that is growing each year. In addition, many of these students are
new immigrants with very limited English language skills.\r\n\r\n     The
majority of our students qualify for free and reduced lunch and are from
85% minority populations. When looking at different indicators, almost al
l of our students are at risk and in need of intervention and support.\r
\n\r\nThis project is for our non English speaking students. Many are str
uggling in their coursework due to the language barrier. With these dicti
onaries, our students will be able to better complete assignments they do
not understand. They will be able to complete work independently by looki
ng up the equivalent words in Spanish for a better understanding of the m
aterial. \r\n\r\nThese dictionaries are also approved for state standardi
zed testing; therefore, students will not only be able to use them in the
classroom but also for testing. The impact projected is better performanc
e in the classroom, testing, and learning the English language.nannan
==================================================
```

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)
```

```
  Many of our students come from single parent and low income homes with
limited transportation. We have a large ESL and special education populat
ion that is growing each year. In addition, many of these students are ne
w immigrants with very limited English language skills.         The major
ity of our students qualify for free and reduced lunch and are from 85% m
inority populations. When looking at different indicators, almost all of
our students are at risk and in need of intervention and support.    This
project is for our non English speaking students. Many are struggling in
their coursework due to the language barrier. With these dictionaries, ou
r students will be able to better complete assignments they do not unders
tand. They will be able to complete work independently by looking up the
equivalent words in Spanish for a better understanding of the material.
These dictionaries are also approved for state standardized testing; ther
efore, students will not only be able to use them in the classroom but al
so for testing. The impact projected is better performance in the classro
om, testing, and learning the English language.nannan
```

In [16]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
```

```
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
 Many of our students come from single parent and low income homes with l
imited transportation We have a large ESL and special education populatio
n that is growing each year In addition many of these students are new im
migrants with very limited English language skills The majority of our st
udents qualify for free and reduced lunch and are from 85 minority popula
tions When looking at different indicators almost all of our students are
at risk and in need of intervention and support This project is for our n
on English speaking students Many are struggling in their coursework due
to the language barrier With these dictionaries our students will be able
to better complete assignments they do not understand They will be able t
o complete work independently by looking up the equivalent words in Spani
sh for a better understanding of the material These dictionaries are also
approved for state standardized testing therefore students will not only
be able to use them in the classroom but also for testing The impact proj
ected is better performance in the classroom testing and learning the Eng
lish language nannan
```

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, 2109.21it/s]
```

In [19]:
```
# after preprocesing
preprocessed_essays_train[20000]
```

Out[19]:
```
'many students come single parent low income homes limited transportation
large esl special education population growing year addition many student
s new immigrants limited english language skills majority students qualif
y free reduced lunch 85 minority populations looking different indicators
almost students risk need intervention support project non english speaki
ng students many struggling coursework due language barrier dictionaries
students able better complete assignments not understand able complete wo
rk independently looking equivalent words spanish better understanding ma
terial dictionaries also approved state standardized testing therefore st
udents not able use classroom also testing impact projected better perfor
mance classroom testing learning english language nannan'
```

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, 2011.00it/s]
```

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

Out[21]:
```
'students extremely poor school title school teach students english secon
d language students arrived native countries students populations consist
following bangladesh lebanon china dominican republic mexico puerto rico
senegal many students lack foundation skills not attended school previous
ly students academic deficits following subjects reading writing math how
ever despite academic deficits students eager learn therefore school comm
itted supporting students families providing school materials notebooks p
encils book bags time develop within self confidence independence enable
students move next level education students use materials class reading w
riting language acquisitions activities post highlighters used writing no
tes reading passages highlighter used highlight supporting details constr
uction paper used stem art projects pencils erasers used writing daily ba
sis rulers used writing math laminating pouches used laminate visual aid
around classroom varies languages window ventana copy paper utilized make
copies students classwork homework stapler paper cutter used class writin
g activities projects 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 [00:11<00:00, 2118.95it/s]
```

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

Out[23]:
```
'11th year teaching however making big move district kindergarten excited
meet smiling faces new five year old babes future students attend school
services many low income students needs far wide however look forward pro
viding safe nurturing exciting classroom environment feel like rock stars
no matter situation home project aimed rewarding students positive social
academic emotional choices sticker stamp quick way make children feel val
ued uplifted choices made simple tangible token way bring positive classr
oom environment life small rewards positive choices encourage students ma
ke positive choices part everything rewarding good academic social emotio
nal choices important building classroom environment full positivity cele
brations trust students higher self esteem praised smallest positive choi
ce biggest project create sense family classroom students light successes
celebrated 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, 141673.36it/s]
```

```
work prek 8 school low ses urban district currently half second graders e
nter grade 3 considered risk reading young students significantly less re
ading experiences peers neighboring communities resulting impact vocabula
ry language use challenge many students collective camaraderie among teac
hers true sense determination improve outcomes students continually refle
ctive looking students respond instruction changing learn keeps motivated
engaged reading writing currently 20 chromebooks not use due lack adequat
e chargers project allow us use technology already order bring increased
personalized learning experiences literacy students opportunity personali
ze literacy experiences classroom select high interest books thereby enga
ging wide reading proven strategy increasing vocabulary may access readin
g software teaches reinforces phonological reading comprehension skills i
nteractive game like application additionally students deeply engaged lit
eracy learning experiences teachers better able engage students tailored
small group instruction 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, 133750.61it/s]
```

```
third grade students live anchorage attend spring hill elementary diverse
school one diversified elementary schools united states school student di
versity includes alaska native korean filipino chinese sudanese hmong ame
rican indian mexican peruvian among beautiful cultures writing goal chall
enge students various engaging activities broaden creativity writing thir
d grade students enthusiastic learners love hands art projects variety pr
ojects engage learning writing academic areas enjoy teaching students var
iety learning opportunities make life long learners pictures earliest for
m communication students gather meaning pictures regardless reading abili
ty native language mind students use watercolors watercolor paper create
high quality pictures write descriptive paragraphs stories using painting
s inspire writing teach students basic elements watercolor painting throu
ghout year students create portfolio include paintings writings seasons w
eather fiction non fiction stories items requesting 16 color prang waterc
olors watercolor paper canvasses materials encourage students produce qua
lity paintings leads quality writing 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, 144391.83it/s]
```

```
technology play huge part lives students future goal program prepare stud
ents 21st century careers exposing much technology possible middle school
s get high school college follow career path suited skills three years ma
ny students class exposed print graphics photography photo manipulation d
igital animation video production computer programming video game design
stop motion animation 3d printing computer aided design robotics classes
made 6th 7th 8th grade students southern california order continue provid
ing students cutting edge technology written grant virtual reality equipm
ent able attain htc vive set sony playstation virtual reality goggles con
sole allow students experience future technology education experiencing t
echnology students continue look future occupations able make informed de
cisions educational futures vista del mar become school continues provide
students unique technological experiences fascinating see students create
future nannan
```

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)
```

```
Quality Art Materials for Quality Kids!!
==================================================
Increased Reading with Leveled Books
==================================================
The Power of Choice: Flexible Seating
==================================================
Help immigrant students succeed in coursework with English-Spanish dictio
naries
==================================================
```

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, 47813.63it/s]
```

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

Out[29]:
```
'it steam teractive part two'
```

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, 48868.02it/s]
```

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

Out[31]:
```
['phenomenon based learning using gadgets gizmos',
 'help us see microscopic world',
 'small group area',
 'working out lessons',
 'toons tiny minds']
```

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, 48435.08it/s]
```

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

Out[33]:
```
['robotics club start up',
 'my school values importance stem education',
 'fight the fidgets',
 'g l a d ly teaching color',
 'cd protection please']
```

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, 1024433.17it/s]
```

```
implementing blended learning
```

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, 1053055.46it/s]
```

```
writing workshop watercolor painting
```

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, 976992.57it/s]
```

```
vr vdm
```

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, 73943.19it/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, 67014.60it/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, 61475.24it/s]
```

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

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

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]:
```
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, 692577.89it/s]
100%|██████████| 24155/24155 [00:00<00:00, 692059.87it/s]
100%|██████████| 36052/36052 [00:00<00:00, 737732.58it/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)
```

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, 2103)
Shape of matrix after one hot encoding  (24155, 2103)
Shape of matrix after one hot encoding  (36052, 2103)
```

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, 2103)
Shape of matrix after one hot encoding  (24155, 2103)
Shape of matrix after one hot encoding  (36052, 2103)
```

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 [03:24, 9399.09it/s]
Done. 1917495  words loaded!
all the words in the coupus 6977096
the unique words in the coupus 43036
The number of words that are present in both glove vectors and our coupus
39278 ( 91.268 %)
word 2 vec length 39278
```

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, 4004.52it/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 [00:05<00:00, 4069.92it/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, 4199.25it/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, 70956.20it/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, 68571.44it/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, 74722.93it/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 [01:19<00:00, 618.05it/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 [00:38<00:00, 622.72it/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 [00:58<00:00, 612.27it/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, 35535.83it/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, 36936.60it/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, 36932.89it/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_ho
t_train,pgc_one_hot_train,teacher_prefix_categories_one_hot_train,categories
_one_hot_train,sub_categories_one_hot_train,prev_projects_train,price_train,
quantity_train)).tocsr()
X_cr = hstack((text_bow_cv,title_bow_cv,school_state_categories_one_hot_cv,p
gc_one_hot_cv,teacher_prefix_categories_one_hot_cv,categories_one_hot_cv,sub
_categories_one_hot_cv,prev_projects_cv,price_cv,quantity_cv)).tocsr()
X_te = hstack((text_bow_test,title_bow_test,school_state_categories_one_hot_
test,pgc_one_hot_test,teacher_prefix_categories_one_hot_test,categories_one_
hot_test,sub_categories_one_hot_test,prev_projects_test,price_test,quantity_
test)).tocsr()

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

```
Final Data matrix
(49041, 7205) (49041,)
(24155, 7205) (24155,)
(36052, 7205) (36052,)
```

SET 2 :

In [79]:
```
X_tr2 = hstack((title_tfidf_train,text_tfidf_train,school_state_categories_o
ne_hot_train,pgc_one_hot_train,teacher_prefix_categories_one_hot_train,categ
ories_one_hot_train,sub_categories_one_hot_train,prev_projects_train,price_t
rain,quantity_train)).tocsr()
X_cr2 = hstack((title_tfidf_cv,text_tfidf_cv,school_state_categories_one_hot
_cv,pgc_one_hot_cv,teacher_prefix_categories_one_hot_cv,categories_one_hot_c
v,sub_categories_one_hot_cv,prev_projects_cv,price_cv,quantity_cv)).tocsr()
X_te2 = hstack((title_tfidf_test,text_tfidf_test,school_state_categories_one
_hot_test,pgc_one_hot_test,teacher_prefix_categories_one_hot_test,categories
_one_hot_test,sub_categories_one_hot_test,prev_projects_test,price_test,quan
tity_test)).tocsr()

print("Final Data matrix")
```

```
print(X_tr2.shape, y_train.shape)
print(X_cr2.shape, y_cv.shape)
print(X_te2.shape, y_test.shape)
```

```
Final Data matrix
(49041, 7205) (49041,)
(24155, 7205) (24155,)
(36052, 7205) (36052,)
```

SET 3 :

In [80]:
```
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 [81]:
```
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 [82]:
```
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 [83]:
```
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 [84]:
```
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 [85]:
```
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 [86]:
```
from scipy.sparse import hstack
X_tr5 = hstack((essay_neg_train,essay_pos_train,essay_neu_train,essay_com_tr
ain,noessay_train,notitle_train,school_state_categories_one_hot_train,pgc_on
e_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_pref
ix_categories_one_hot_cv,categories_one_hot_cv,sub_categories_one_hot_cv,pre
v_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_t
est,teacher_prefix_categories_one_hot_test,categories_one_hot_test,sub_categ
ories_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 7: SVM

1. **[Task-1] Apply Support Vector Machines(SGDClassifier with hinge loss: Linear SVM) on these feature sets**

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

- Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
- Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
- Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

2. **The hyper paramter tuning (best alpha in range [10^-4 to 10^4], and the best penalty among 'l1', 'l2')**

   - 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 the Support Vector Machines on these features by finding the best hyper paramter as suggested in step 2 and step 3**

   - 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
     - **Apply TruncatedSVD on TfidfVectorizer of essay text, choose the number of components (`n_components`) using elbow method** : numerical data

   - **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.

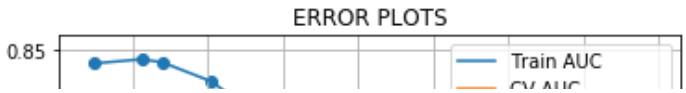
Support Vector Machines

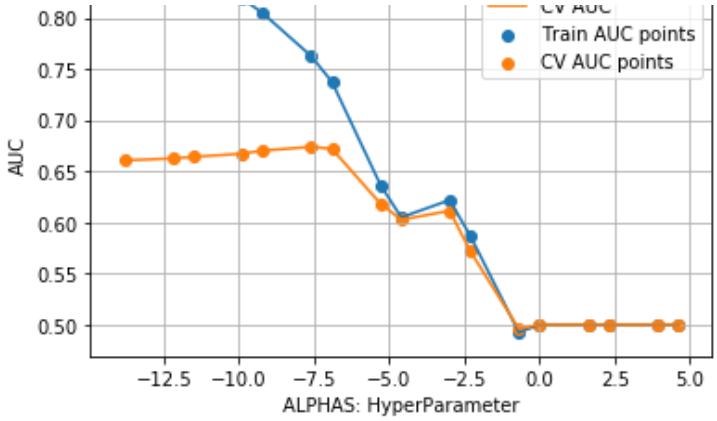
SVM(SGD Classifier With Hinge Loss) Linear SVM On SET 1 :

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

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

alphas = [0.000001,0.000005,0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100]

for i in tqdm(alphas):
    lr = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=i , cl
ass_weight='balanced')
    lr.fit(X_tr, y_train)
    calibrated = CalibratedClassifierCV(lr, method='sigmoid', cv=5)
    calibrated.fit(X_tr, y_train)
    y_train_pred=calibrated.predict_proba(X_tr)[:, 1]
    y_cv_pred=calibrated.predict_proba(X_cr)[:, 1]
    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 [00:34<00:00,  1.74s/it]
```

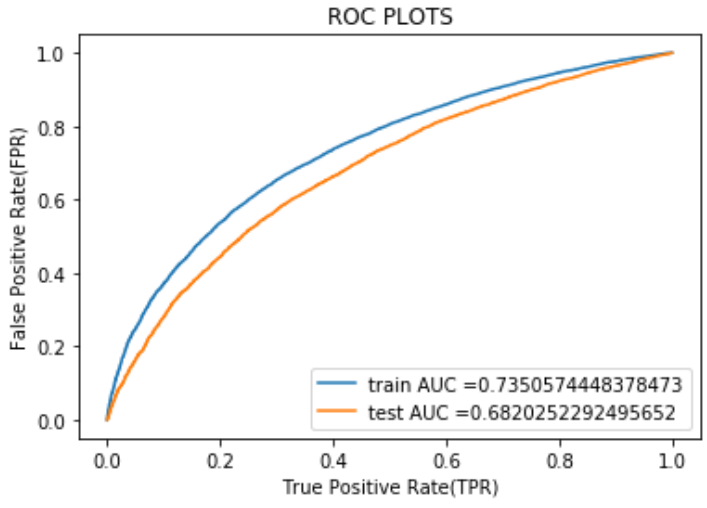
In [88]:
```
import numpy
plt.plot(numpy.log(alphas), train_auc, label='Train AUC')
plt.plot(numpy.log(alphas), cv_auc, label='CV AUC')

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

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





In [89]:
```
best_alpha = 0.001
```

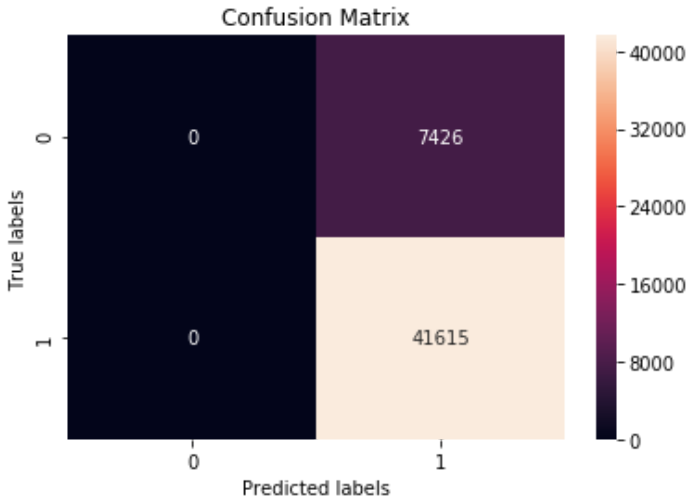
In [90]:
```
from sklearn.metrics import roc_curve, auc
ne = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=best_alpha
, class_weight='balanced')
ne.fit(X_tr,y_train)
calibrated = CalibratedClassifierCV(ne, method='sigmoid', cv=5)
calibrated.fit(X_tr, y_train)
train_fpr, train_tpr, thresholds = roc_curve(y_train, calibrated.predict_pro
ba(X_tr)[:, 1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, calibrated.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)
```



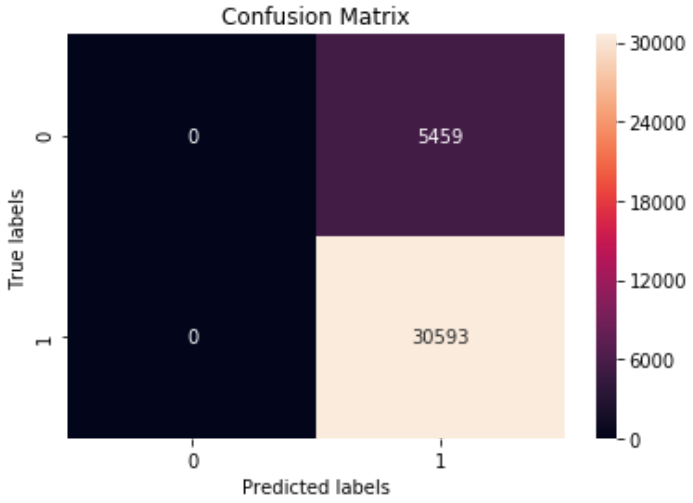
==========

CONFUSION MATRIX :

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

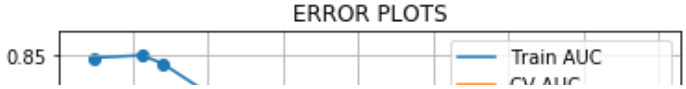


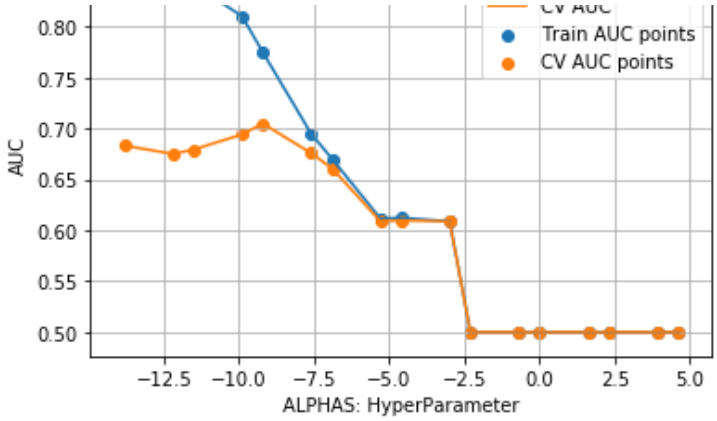
SVM(SGD Classifier With Hinge Loss) Linear SVM On SET 2 :

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

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

alphas = [0.000001,0.000005,0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100]

for i in tqdm(alphas):
    lr = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=i , cl
ass_weight='balanced')
    lr.fit(X_tr2, y_train)
    calibrated = CalibratedClassifierCV(lr, method='sigmoid', cv=5)
    calibrated.fit(X_tr2, y_train)
    y_train_pred=calibrated.predict_proba(X_tr2)[:, 1]
    y_cv_pred=calibrated.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 [00:32<00:00,  1.73s/it]
```

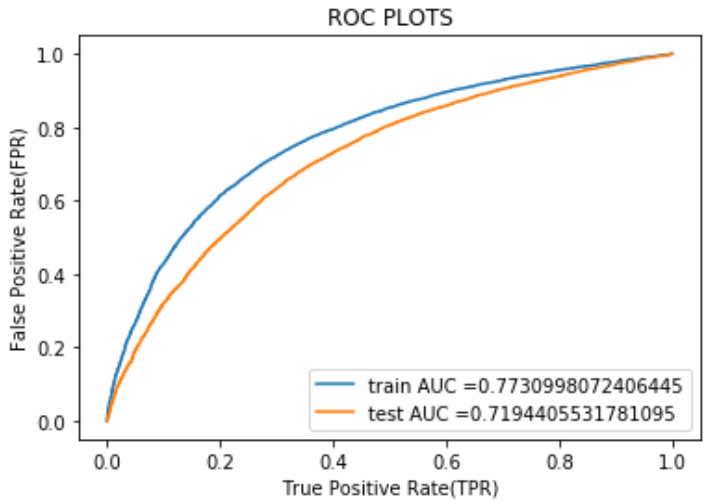
In [94]:
```
import numpy
plt.plot(numpy.log(alphas), train_auc, label='Train AUC')
plt.plot(numpy.log(alphas), cv_auc, label='CV AUC')

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

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





In [95]:
```
best_alpha = 0.0001
```

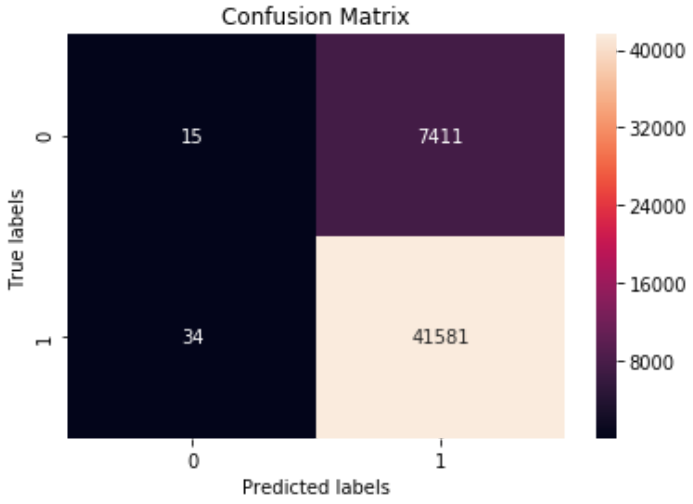
In [96]:
```
from sklearn.metrics import roc_curve, auc
ne = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=best_alpha
, class_weight='balanced')
ne.fit(X_tr2,y_train)
calibrated = CalibratedClassifierCV(ne, method='sigmoid', cv=5)
calibrated.fit(X_tr2, y_train)
# roc_auc_score(y_true, 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, calibrated.predict_pro
ba(X_tr2)[:, 1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, calibrated.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)
```



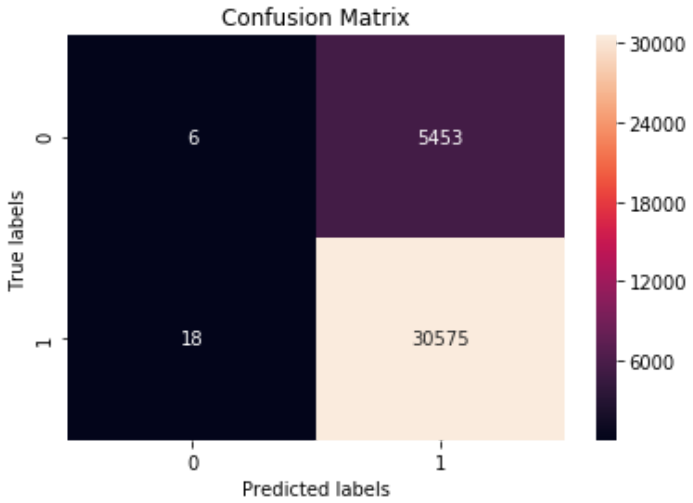
==========
==========

CONFUSION MATRIX :

In [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_train, calibrated.predict(X_tr2)), annot=True, ax = ax,fmt='g');

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



In [98]:
```
#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, calibrated.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');
```

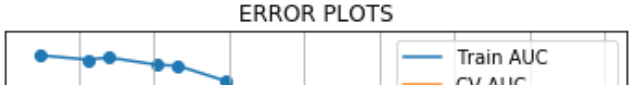


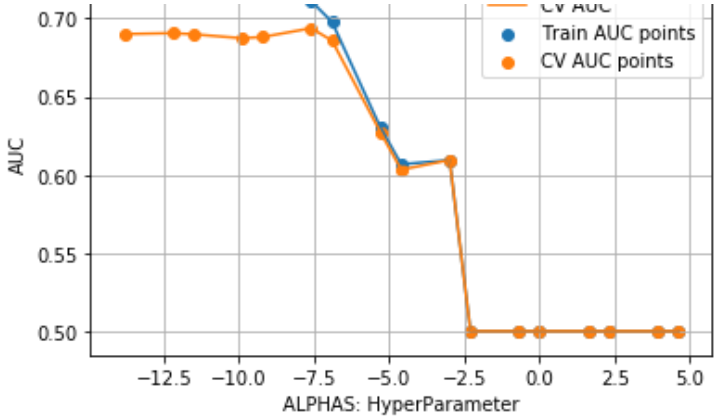
SVM(SGD Classifier With Hinge Loss) Linear SVM On SET 3 :

In [99]:
```
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import roc_auc_score

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

alphas = [0.000001,0.000005,0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100]

for i in tqdm(alphas):
    lr = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=i , cl
ass_weight='balanced')
    lr.fit(X_tr3, y_train)
    calibrated = CalibratedClassifierCV(lr, method='sigmoid', cv=5)
    calibrated.fit(X_tr3, y_train)
    y_train_pred=calibrated.predict_proba(X_tr3)[:, 1]
    y_cv_pred=calibrated.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%|██████████| 17/17 [02:23<00:00,  9.18s/it]
```

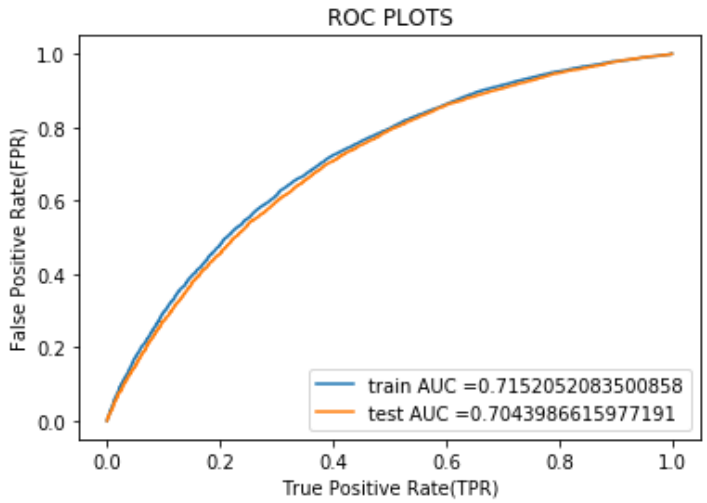
In [100]:
```
import numpy
plt.plot(numpy.log(alphas), train_auc, label='Train AUC')
plt.plot(numpy.log(alphas), cv_auc, label='CV AUC')

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

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





In [101]:
```
best_alpha = 0.0005
```

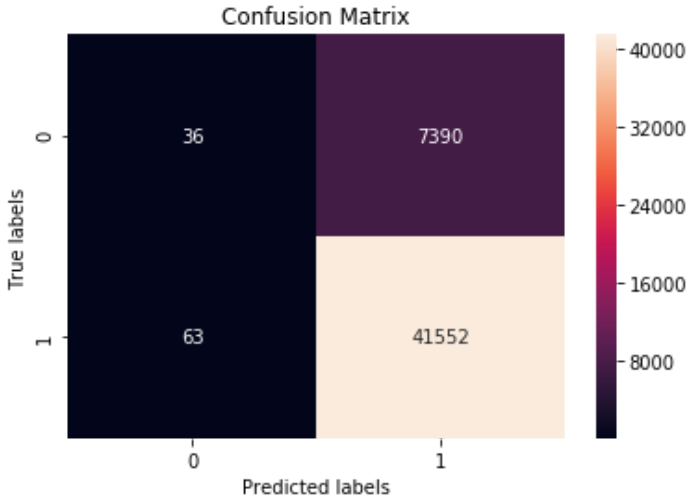
In [102]:
```
from sklearn.metrics import roc_curve, auc
ne = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=best_alpha
, class_weight='balanced')
ne.fit(X_tr3,y_train)
calibrated = CalibratedClassifierCV(ne, method='sigmoid', cv=5)
calibrated.fit(X_tr3, y_train)
# roc_auc_score(y_true, 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, calibrated.predict_pro
ba(X_tr3)[:, 1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, calibrated.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)
```



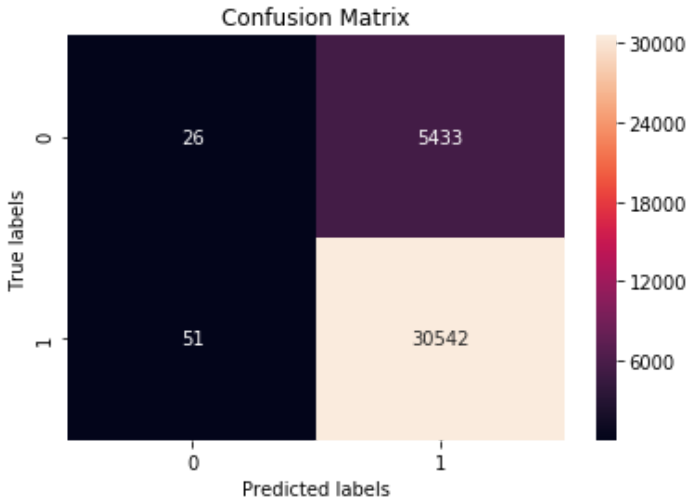
==========
==========

CONFUSION MATRIX :

In [103]:
```
#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, calibrated.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 [104]:
```
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, calibrated.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');
```

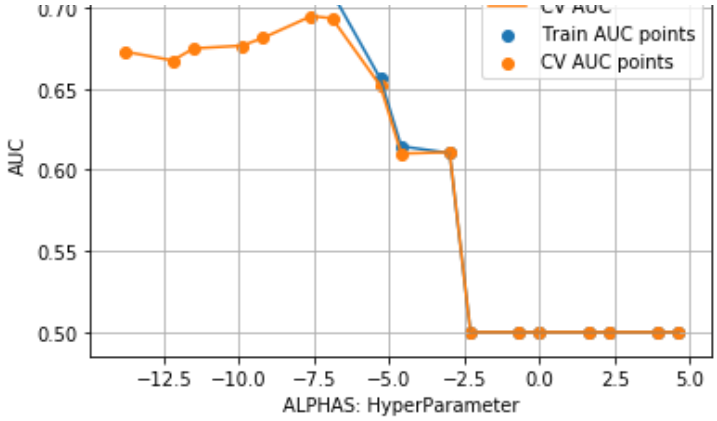


SVM(SGD Classifier With Hinge Loss) Linear SVM On SET 4 :

In [105]:
```
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import roc_auc_score

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

alphas = [0.000001,0.000005,0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100]

for i in tqdm(alphas):
    lr = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=i , cl
ass_weight='balanced')
    lr.fit(X_tr4, y_train)
    calibrated = CalibratedClassifierCV(lr, method='sigmoid', cv=5)
    calibrated.fit(X_tr4, y_train)
    y_train_pred=calibrated.predict_proba(X_tr4)[:, 1]
    y_cv_pred=calibrated.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))
```

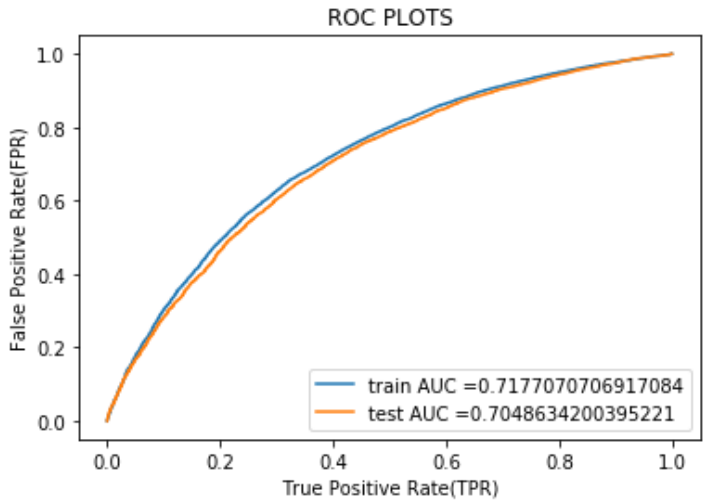
```
100%|██████████| 17/17 [02:25<00:00,  9.05s/it]
```

In [106]:
```
import numpy
plt.plot(numpy.log(alphas), train_auc, label='Train AUC')
plt.plot(numpy.log(alphas), cv_auc, label='CV AUC')

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

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



In [107]:
```
best_alpha = 0.0005
```

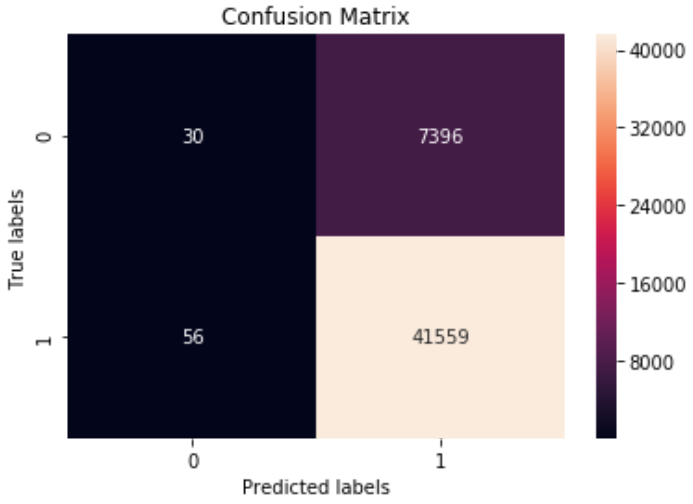
In [108]:
```
from sklearn.metrics import roc_curve, auc
ne = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=best_alpha
, class_weight='balanced')
ne.fit(X_tr4,y_train)
calibrated = CalibratedClassifierCV(ne, method='sigmoid', cv=5)
calibrated.fit(X_tr4, y_train)
# roc_auc_score(y_true, 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, calibrated.predict_pro
ba(X_tr4)[:, 1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, calibrated.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)
```



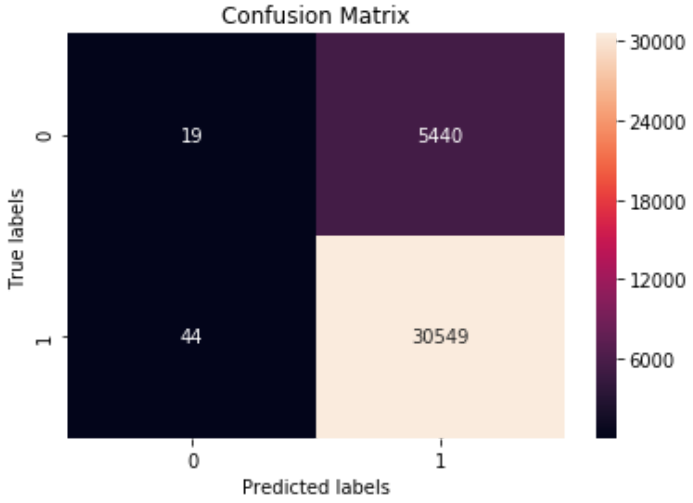
==========
==========

CONFUSION MATRIX :

In [109]:
```
#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, calibrated.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 [110]:
```
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, calibrated.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');
```



Decomposition Of TFIDF Vectorizer :

In [111]:
```
from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components=100, random_state=42, algorithm='randomized'
)
svd.fit(text_tfidf_train)
svdtfidftrain=svd.transform(text_tfidf_train)
svdtfidfcv=svd.transform(text_tfidf_cv)
svdtfidftest=svd.transform(text_tfidf_test)
```

In [112]:
```
X_tr5 = hstack((X_tr5,svdtfidftrain)).tocsr()
X_cr5 = hstack((X_cr5,svdtfidfcv)).tocsr()
X_te5 = hstack((X_te5,svdtfidftest)).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)
```

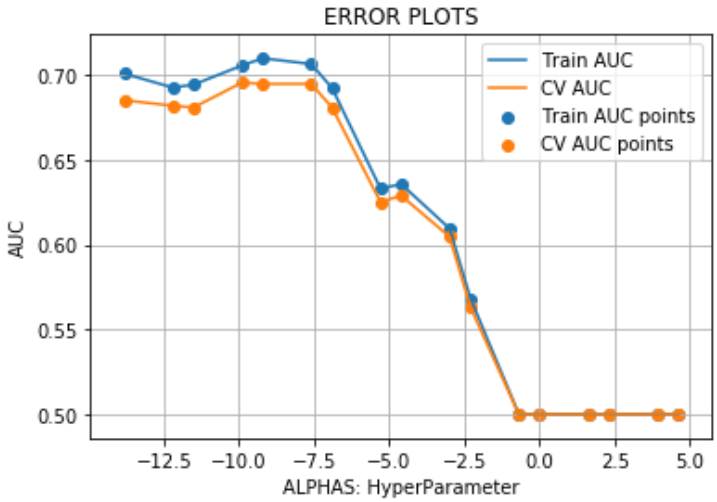
```
Final Data matrix
(49041, 208) (49041,)
(24155, 208) (24155,)
(36052, 208) (36052,)
```

In [113]:
```
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import roc_auc_score

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

alphas = [0.000001,0.000005,0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100]

for i in tqdm(alphas):
    lr = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=i , cl
ass_weight='balanced')
    lr.fit(X_tr5, y_train)
    calibrated = CalibratedClassifierCV(lr, method='sigmoid', cv=5)
    calibrated.fit(X_tr5, y_train)
    y_train_pred=calibrated.predict_proba(X_tr5)[:, 1]
    y_cv_pred=calibrated.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 [00:31<00:00,  2.01s/it]
```

In [114]:
```
import numpy
plt.plot(numpy.log(alphas), train_auc, label='Train AUC')
plt.plot(numpy.log(alphas), cv_auc, label='CV AUC')

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

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

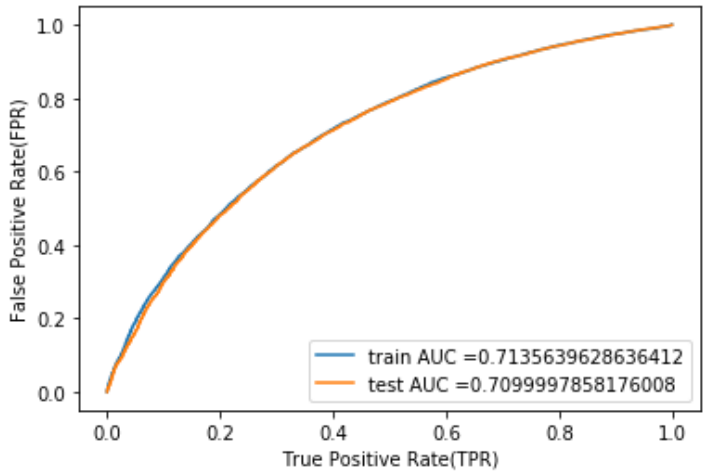
```
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [115]:
```
best_alpha = 0.0001
```

In [116]:
```
from sklearn.metrics import roc_curve, auc
ne = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=best_alpha
, class_weight='balanced')
ne.fit(X_tr5,y_train)
calibrated = CalibratedClassifierCV(ne, method='sigmoid', cv=5)
calibrated.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, calibrated.predict_pro
ba(X_tr5)[:, 1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, calibrated.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)
```

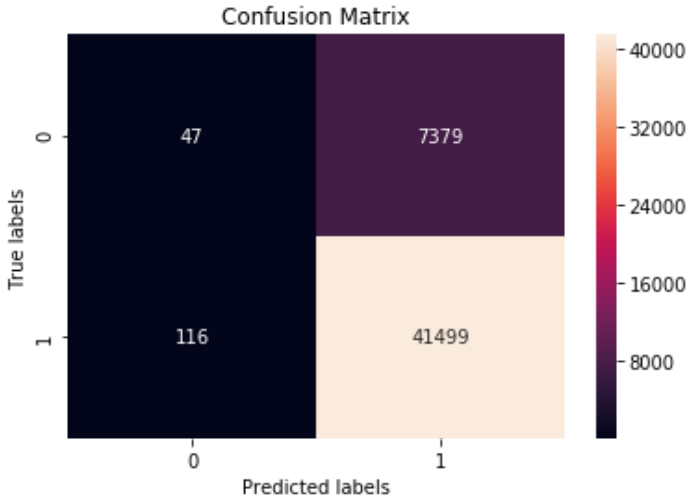
ROC PLOTS



==========

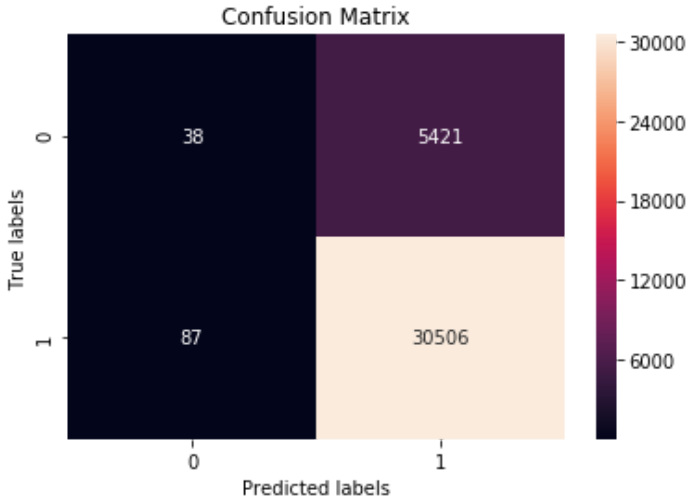
CONFUSION MATRIX :

In [117]:
```
#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, calibrated.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 [118]:
```
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, calibrated.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');
```



USING SKLEARN'S KERNEL APPROXIMATION NYSTROEM :

In [119]:
```
from sklearn.kernel_approximation import Nystroem
nys = Nystroem(kernel='rbf', random_state=42, n_components=800)
nys.fit(X_tr5)
nystrain=nys.transform(X_tr5)
nyscv=nys.transform(X_cr5)
nystest=nys.transform(X_te5)
print("Final Data matrix")
print(nystrain.shape, y_train.shape)
print(nyscv.shape, y_cv.shape)
print(nystest.shape, y_test.shape)
```

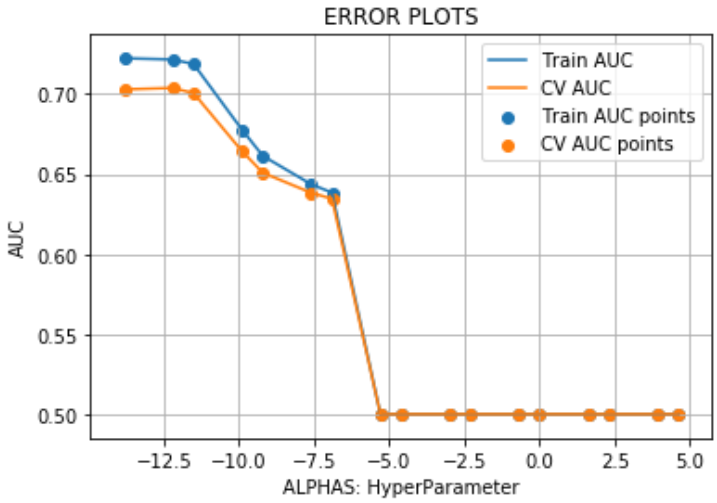
```
Final Data matrix
(49041, 800) (49041,)
(24155, 800) (24155,)
(36052, 800) (36052,)
```

In [120]:
```
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import roc_auc_score

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

alphas = [0.000001,0.000005,0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005,
0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100]

for i in tqdm(alphas):
    lr = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=i , cl
ass_weight='balanced')
    lr.fit(nystrain, y_train)
    calibrated = CalibratedClassifierCV(lr, method='sigmoid', cv=5)
    calibrated.fit(nystrain, y_train)
    y_train_pred=calibrated.predict_proba(nystrain)[:, 1]
    y_cv_pred=calibrated.predict_proba(nyscv)[:, 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 [01:46<00:00,  6.21s/it]
```

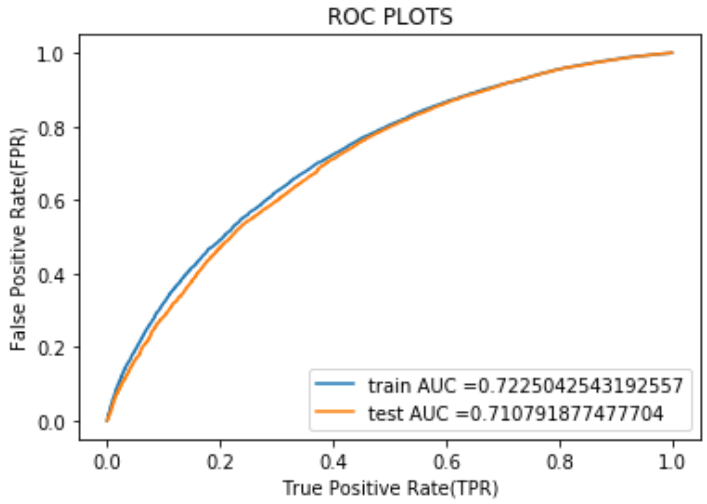
In [121]:
```
import numpy
plt.plot(numpy.log(alphas), train_auc, label='Train AUC')
plt.plot(numpy.log(alphas), cv_auc, label='CV AUC')

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

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



In [122]:
```
best_alpha = 0.000005
```

In [123]:
```
from sklearn.metrics import roc_curve, auc
ne = linear_model.SGDClassifier(loss='hinge', penalty='l1', alpha=best_alpha
, class_weight='balanced')
ne.fit(nystrain,y_train)
calibrated = CalibratedClassifierCV(ne, method='sigmoid', cv=5)
calibrated.fit(nystrain, 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, calibrated.predict_pro
ba(nystrain)[:, 1])
test_fpr, test_tpr, thresholds = roc_curve(y_test, calibrated.predict_proba(
nystest)[:, 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")
```

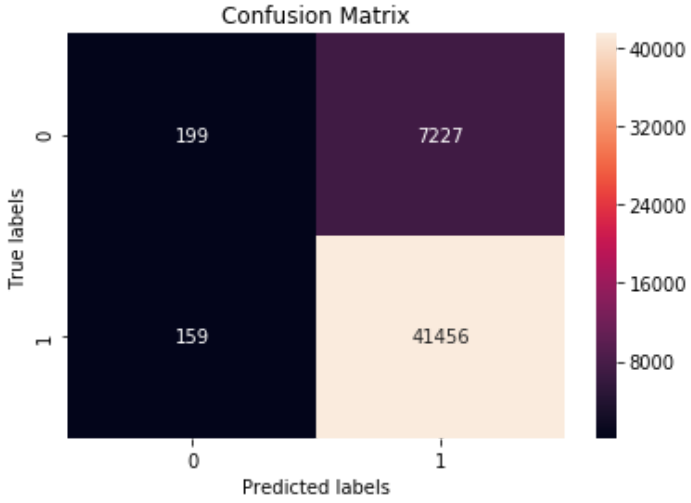
```
plt.show()
print("="*100)
```



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

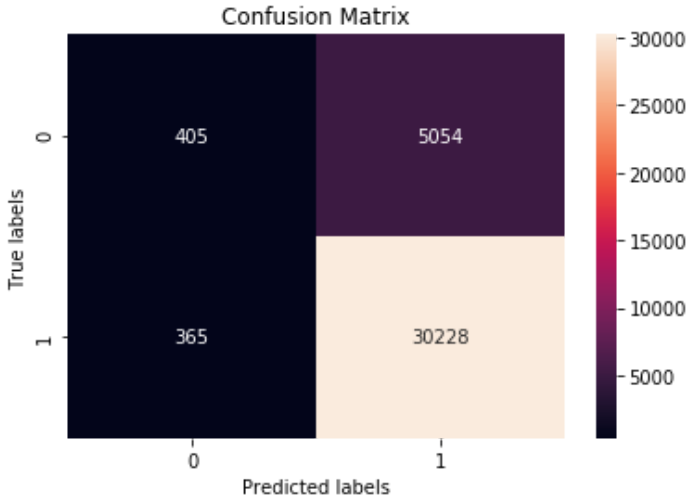
CONFUSION MATRIX :

In [124]:
```
#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, calibrated.predict(nystrain)), 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 [125]:
```
import seaborn as sns
import matplotlib.pyplot as plt
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, calibrated.predict(nystest)), 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 [126]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Hyper Parameter", "AUC"]

x.add_row(["BOW(USING STANDARD SCALER)",0.001,68.7])
x.add_row(["TFIDF(USING STANDARD SCALER)",0.0001,71.3])
x.add_row(["AVG W2V(USING STANDARD SCALER)",0.0005,69.3])
x.add_row(["TFIDF W2V(USING STANDARD SCALER)",0.0005,70.3])
x.add_row(["NO OF WORDS/SENTIMENT SCORES",0.0001,70.3])
x.add_row(["NO OF WORDS/SENTIMENT SCORES(kernel approx)",0.000005,70.8])
print(x.get_string(titles = "Logistic Regression - Observations"))
```

```
+-----------------------------------------+-----------------+------+
|                Vectorizer               | Hyper Parameter | AUC  |
+-----------------------------------------+-----------------+------+
|       BOW(USING STANDARD SCALER)        |      0.001      | 68.7 |
|      TFIDF(USING STANDARD SCALER)       |      0.0001     | 71.3 |
|     AVG W2V(USING STANDARD SCALER)      |      0.0005     | 69.3 |
|    TFIDF W2V(USING STANDARD SCALER)     |      0.0005     | 70.3 |
|      NO OF WORDS/SENTIMENT SCORES       |      0.0001     | 70.3 |
| NO OF WORDS/SENTIMENT SCORES(kernel approx) |      5e-06      | 70.8 |
+-----------------------------------------+-----------------+------+
```