

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
<code>project_id</code>	A unique identifier for the proposed project. Example: p036502
<code>project_title</code>	Title of the project. Examples: <ul style="list-style-type: none">• Art Will Make You Happy!• First Grade Fun
<code>project_grade_category</code>	Grade level of students for which the project is targeted. One of the following enumerated values: <ul style="list-style-type: none">• Grades PreK-2• Grades 3-5• Grades 6-8• Grades 9-12
<code>project_subject_categories</code>	One or more (comma-separated) subject categories for the project from the following enumerated list of values: <ul style="list-style-type: none">• Applied Learning• Care & Hunger• Health & Sports• History & Civics• Literacy & Language• Math & Science• Music & The Arts• Special Needs• Warmth Examples: <ul style="list-style-type: none">• Music & The Arts• Literacy & Language, Math & Science
<code>school_state</code>	State where school is located (Two-letter U.S. postal code). Example: WY
<code>project_subject_subcategories</code>	One or more (comma-separated) subject subcategories for the project. Examples: <ul style="list-style-type: none">• Literacy• Literature & Writing, Social Sciences

Feature	Description
<code>project_resource_summary</code>	Description of the resources needed for the project. Example: <ul style="list-style-type: none"> My students need hands on literacy materials to manage sensory needs!
<code>project_essay_1</code>	First application essay*
<code>project_essay_2</code>	Second application essay*
<code>project_essay_3</code>	Third application essay*
<code>project_essay_4</code>	Fourth application essay*
<code>project_submitted_datetime</code>	Datetime when project application was submitted. Example: 2016-04-28 12:43:56.245
<code>teacher_id</code>	A unique identifier for the teacher of the proposed project. Example: bdf8baa8fedef6bfeec7ae4ff1c15c56
<code>teacher_prefix</code>	Teacher's title. One of the following enumerated values: <ul style="list-style-type: none"> nan Dr. Mr. Mrs. Ms. Teacher.
<code>teacher_number_of_previously_posted_projects</code>	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
<code>id</code>	A <code>project_id</code> value from the <code>train.csv</code> file. Example: p036502
<code>description</code>	Description of the resource. Example: Tenor Saxophone Reeds, Box of 25
<code>quantity</code>	Quantity of the resource required. Example: 3
<code>price</code>	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
<code>project_is_approved</code>	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 [181]:

```
%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
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
```

1.1 Reading Data

In [182]:

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

In [183]:

```
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 [184]:

```
project_data=project_data[pd.notnull(project_data['teacher_prefix'])]
```

In [185]:

```
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[185]:

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

1.2 preprocessing of project_subject_categories

In [186]:

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

cat_list = []
for i in categories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the category 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 placing all the ' ' (space) with '' (empty) ex: "Math & Science"=> "Math&Science"
            temp+=j.strip()+" " # " abc ".strip() will return "abc", remove the trailing spaces
            temp = temp.replace('&', '_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of project_subject_subcategories

In [187]:

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

sub_cat_list = []
for i in sub_categories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the category based on space "Math & Science"=>
            "Math", "&", "Science"
            i=i.replace('The', '') # if we have the words "The" we are going to replace it with '' (i.e removing 'The')
```

```

removing 'The')
j = j.replace(' ', '') # we are placing all the ' ' (space) with '' (empty) ex: "Math & Science" =>
"Math&Science"
temp += j.strip() + " #" + abc ".strip() will return "abc", remove the trailing spaces
temp = temp.replace('&', '_')
sub_cat_list.append(temp.strip())

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

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

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

```

1.3 Text preprocessing

In [188]:

```

# 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 [189]:

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

Out[189]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bcd1151f324dd63a	Mr.	FL	2016-10-25 09:22:10

In [108]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [190]:

```

# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])

```

```
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

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

The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes the students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard work put in during the school year, with a dunk tank being the most popular activity. My students will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as special chairs students will each use on occasion. I will utilize them in place of chairs at my small group tables during math and reading times. The rest of the day they will be used by the students who need the highest amount of movement in their life in order to stay focused on school. \r\n\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. When the students are sitting in group with me on the Hokki Stools, they are always moving, but at the same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be taken. There are always students who head over to the kidney table to get one of the stools who are disappointed as there are not enough of them. \r\n\r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the same time. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit still. nannan

How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room for my students look forward to coming to each day.

My class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas. They attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an "open classroom" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more. With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.

Your generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.

It costs a lot of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan

My wonderful students are 3, 4, and 5 years old. We are located in a small town outside of Charlotte, NC. All of my 22 students are children of school district employees.\r\nMy students are bright, energetic, and they love to learn! They love hands-on activities that get them moving. Like most preschoolers, they enjoy music and creating different things. \r\nAll of my students come from wonderful families that are very supportive of our classroom. Our parents enjoy watching their children's growth as much as we do! These materials will help me teach my students all about the life cycle of a butterfly. We will watch as the Painted Lady caterpillars grow bigger and build their chrysalis. After a few weeks they will emerge from the chrysalis as beautiful butterflies! We already have a net for the chrysalises, but we still need the caterpillars and feeding station.\r\nThis will be an unforgettable experience for my students. My student absolutely love hands-on materials. They learn so much from getting to touch and manipulate different things. The supporting materials I have selected will help my students understand the life cycle through exploration.nannan

=====

The students in my classroom are learners, readers, writers, explorers, scientists, and mathematicians! The potential in these first graders is endless! Each day they come in grinning from ear-to-ear and ready to learn more. \r\nI choose curriculum that is real and relevant to the students, but it will also prepare them for their futures. These kids are encouraged to investigate concepts that are exciting for them and I hope we can keep this momentum going! These kids deserve the best, please help me give that to them! Thank you! :)These kits include a wide variety of science, technology, engineering, and mechanics for my students to dive into at the beginning of the year. I want them to hit the ground running this upcoming year and these kits always encourage high interest.\r\nWho wouldn't want to build their own roller coaster, design a car, or even think critically to make a bean bag bounce as far as it can go?? These kits will also show students potential careers that they may have never heard of before!\r\nAny donations would be greatly appreciated and my students will know exactly who to thank for them!\nannan

=====

In [191]:

```
# 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 [192]:

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

My wonderful students are 3, 4, and 5 years old. We are located in a small town outside of Charlotte, NC. All of my 22 students are children of school district employees.\r\nMy students are bright, energetic, and they love to learn! They love hands-on activities that get them moving. Like most preschoolers, they enjoy music and creating different things. \r\nAll of my students come from wonderful families that are very supportive of our classroom. Our parents enjoy watching their children's growth as much as we do!These materials will help me teach my students all about the life cycle of a butterfly. We will watch as the Painted Lady caterpillars grow bigger and build their chrysalis. After a few weeks they will emerge from the chrysalis as beautiful butterflies! We already have a net for the chrysalises, but we still need the caterpillars and feeding station.\r\nThis will be an unforgettable experience for my students. My student absolutely love hands-on materials. They learn so much from getting to touch and manipulate different things. The supporting materials I have selected will help my students understand the life cycle through exploration.\nannan

=====

In [193]:

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

My wonderful students are 3, 4, and 5 years old. We are located in a small town outside of Charlotte, NC. All of my 22 students are children of school district employees. My students are bright, energetic, and they love to learn! They love hands-on activities that get them moving. Like most preschoolers, they enjoy music and creating different things. All of my students come from wonderful families that are very supportive of our classroom. Our parents enjoy watching their children's growth as much as we do!These materials will help me teach my students all about the life cycle of a butterfly. We will watch as the Painted Lady caterpillars grow bigger and build their chrysalis. After a few weeks they will emerge from the chrysalis as beautiful butterflies! We already have a net for the chrysalises, but we still need the caterpillars and feeding station. This will be an unforgettable experience for my students. My student absolutely love hands-on materials. They learn so much from getting to touch and m

manipulate different things. The supporting materials I have selected will help my students understand the life cycle through exploration.nannan

In [194]:

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

My wonderful students are 3 4 and 5 years old We are located in a small town outside of Charlotte NC All of my 22 students are children of school district employees My students are bright energetic and they love to learn They love hands on activities that get them moving Like most preschoolers they enjoy music and creating different things All of my students come from wonderful families that are very supportive of our classroom Our parents enjoy watching their childrens growth as much as we do These materials will help me teach my students all about the life cycle of a butterfly We will watch as the Painted Lady caterpillars grow bigger and build their chrysalis After a few weeks they will emerge from the chrysalis as beautiful butterflies We already have a net for the chrysalises but we still need the caterpillars and feeding station This will be an unforgettable experience for my students My student absolutely loves hands on materials They learn so much from getting to touch and manipulate different things The supporting materials I have selected will help my students understand the life cycle through exploration nannan

In [196]:

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

In [199]:

```
# Combining all the above students
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['essay'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\r', ' ')
    sent = sent.replace('\n', ' ')
    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 not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```


In [200]:

```
# after preprocessing
preprocessed_essays[20000]
```

Out[200]:

'my wonderful students 3 4 5 years old we located small town outside charlotte nc all 22 students children school district employees my students bright energetic love learn they love hands activities get moving like preschoolers enjoy music creating different things all students come wonderful families supportive classroom our parents enjoy watching children growth much these materials help teach students life cycle butterfly we watch painted lady caterpillars grow bigger build chrysalis after weeks emerge chrysalis beautiful butterflies we already net chrysalises still need caterpillars feeding station this unforgettable experience students my student absolutely love hands materials they learn much getting touch manipulate different things the supporting materials i selected help students understand life cycle exploration nannan'

1.4 Preprocessing of `project_title`

In [201]:

```
# similarly you can preprocess the titles also
from tqdm import tqdm
preprocessed_title = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['project_title'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\n', ' ')
    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 not in stopwords)
    preprocessed_title.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109245/109245 [00:06<00:00, 16186.57it/s]
```

1.5 Preparing data for models

In [202]:

```
project data.columns
```

Out[202]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
      'project_submitted_datetime', 'project_grade_category', 'project_title',
      'project_essay_1', 'project_essay_2', 'project_essay_3',
      'project_essay_4', 'project_resource_summary',
      'teacher_number_of_previously_posted_projects', 'project_is_approved',
      '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 (optional)

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

1.5.1 Vectorizing Categorical data

In [119]:

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

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sp
orts', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig (109245, 9)
```

In [120]:

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

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_
Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'Perf
ormingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geogr
aphy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualA
rts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Liter
acy']
Shape of matrix after one hot encodig (109245, 30)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [122]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

Shape of matrix after one hot encodig (109245, 16623)

1.5.2.2 TFIDF vectorizer

In [124]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

Shape of matrix after one hot encodig (109245, 16623)

1.5.2.3 Using Pretrained Models: Avg W2V

In [125]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
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')

# =====
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495 words loaded!

# =====

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    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 coupus", \
      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/how-to-use-pickle-to-save-an
d-load-variables-in-python/

import pickle
with open('glove_vectors', 'wb') as f:
    pickle.dump(words_courpus, f)

'''
```

Out[125]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef loadGloveModel(\ngloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\r', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n\n# =====\n\nOutput:\n\nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495 words loaded!\n\n# =====\n\n\nwords = []\nfor i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    words.extend(i.split(\' \'))\n\nprint("all the words in the coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words that are present in both glove vectors and our coupus", \n      len(inter_words), "(" ,np.round(len(inter_words)/len(words)*100,3), "%")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n\n\n\n# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n\n'
```

In [126]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-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())
```

In [127]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # 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.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109245/109245 [01:13<00:
00, 1485.05it/s]
```

109245
300

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [128]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# 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 [129]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # 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/review
    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.split())) # 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.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

100% | ██████████ 100245 / 100245 [09.44/00

```
100%|████████████████████████████████████████████████████████████████████████████████| 109245/109245 [00:44<00:00, 208.22it/s]
```

```
109245
300
```

In [130]:

```
# Similarly you can vectorize for title also
```

1.5.3 Vectorizing Numerical features

In [203]:

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

In [204]:

```
# check this one: https://www.youtube.com/watch?v=0H0qOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

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

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

```
Mean : 298.1152448166964, Standard deviation : 367.49642545627506
```

In [205]:

```
price_standardized
```

Out[205]:

```
array([[ -0.39052147],
       [  0.00240752],
       [  0.5952024 ],
       ...,
       [-0.1582471 ],
       [-0.61242839],
       [-0.51215531]])
```

1.5.4 Merging all the above features

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

In [134]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(109245, 9)
(109245, 30)
```

```
(109245, 0),  
(109245, 16623)  
(109245, 1)
```

In [135]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039  
from scipy.sparse import hstack  
# with the same hstack function we are concatenating a sparse matrix and a dense matrix :)  
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))  
X.shape
```

Out[135]:

```
(109245, 16663)
```

Computing Sentiment Scores

In [327]:

```
import nltk  
from nltk.sentiment.vader import SentimentIntensityAnalyzer  
  
import nltk  
nltk.download('vader_lexicon')  
  
sid = SentimentIntensityAnalyzer()  
  
for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students with the  
biggest enthusiasm \\  
for learning my students learn in many different ways using all of our senses and multiple intelligence  
s i use a wide range \\  
of techniques to help all my students succeed students in my class come from a variety of different bac  
kgrounds which makes \\  
for wonderful sharing of experiences and cultures including native americans our school is a caring com  
munity of successful \\  
learners which can be seen through collaborative student project based learning in and out of the class  
room kindergarteners \\  
in my class love to work with hands on materials and have many different opportunities to practice a sk  
ill before it is \\  
mastered having the social skills to work cooperatively with friends is a crucial aspect of the kinderg  
arten curriculum \\  
montana is the perfect place to learn about agriculture and nutrition my students love to role play in  
our pretend kitchen \\  
in the early childhood classroom i have had several kids ask me can we try cooking with real food i wil  
l take their idea \\  
and create common core cooking lessons where we learn important math and writing concepts while cooking  
delicious healthy \\  
food for snack time my students will have a grounded appreciation for the work that went into making th  
e food and knowledge \\  
of where the ingredients came from as well as how it is healthy for their bodies this project would exp  
and our learning of \\  
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce  
make our own bread \\  
and mix up healthy plants from our classroom garden in the spring we will also create our own cookbooks  
to be printed and \\  
shared with families students will gain math and literature skills as well as a life long enjoyment for  
healthy cooking \\  
nannan'  
ss = sid.polarity_scores(for_sentiment)  
  
for k in ss:  
    print('{0}: {1}, '.format(k, ss[k]), end='')  
  
# we can use these 4 things as features/attributes (neg, neu, pos, compound)  
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
[nltk_data] Downloading package vader_lexicon to  
[nltk_data] C:\Users\om\AppData\Roaming\nltk_data...  
[nltk_data] Package vader_lexicon is already up-to-date!  
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

1. Logistic Regression(either SGDClassifier with log loss) on these feature sets

- **Set 1:** categorical, numerical features + project_title(BOW) + preprocessed_eassay (`BOW with bi-grams` with `min_df=10` and `max_features=5000`)
- **Set 2:** categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (`TFIDF with bi-grams` with `min_df=10` and `max_features=5000`)
- **Set 3:** categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
- **Set 4:** categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)

2. 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

3. Conclusion

- Summary of the Results Obtained

2. Logistic Regression

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

In [206]:

```
project_data.drop('essay',axis=1,inplace=True)
```

In [207]:

```
project_data['essay']=preprocessed_essays  
project_data['title']=preprocessed_title
```

In [208]:

```
y=project_data['project_is_approved']  
project_data.drop('project_is_approved',axis=1,inplace=True)
```

In [245]:

```
print(y.value_counts())
```

```
1    92703  
0    16542  
Name: project_is_approved, dtype: int64
```

In [209]:

```
from sklearn.model_selection import train_test_split  
from sklearn.linear_model import LogisticRegression  
from sklearn.datasets import *  
  
x_train,x_test,y_train,y_test=train_test_split(project_data,y,test_size=0.33,stratify=y)  
x_train,x_cv,y_train,y_cv=train_test_split(x_train,y_train,test_size=0.33,stratify=y_train)  
  
print("Shape of Train Data is {}".format(x_train.shape,y_train.shape))  
print("Shape of CV Data is {}".format(x_cv.shape,y_cv.shape))
```

```
print("Shape of Test Data is {}".format(x_test.shape,y_test.shape))
```

```
project_data
```

Shape of Train Data is (49039, 20)

Shape of CV Data is (24155, 20)

Shape of Test Data is (36051, 20)

Out[209]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_da
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56
3	45	p246581	f3cb9bffba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09
5	141660	p154343	a50a390e8327a95b77b9e495b58b9a6e	Mrs.	FL	2017-04-08 22:40:43
6	21147	p099819	9b40170bfa65e399981717ee8731efc3	Mrs.	CT	2017-02-17 19:58:56
7	94142	p092424	5bfd3d12fae3d2fe88684bbac570c9d2	Ms.	GA	2016-09-01 00:02:15
8	112489	p045029	487448f5226005d08d36bdd75f095b31	Mrs.	SC	2016-09-25 17:00:26
9	158561	p001713	140eeac1885c820ad5592a409a3a8994	Ms.	NC	2016-11-17 18:18:56

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_da
10	43184	p040307	363788b51d40d978fe276bcb1f8a2b35	Mrs.	CA	2017-01-04 16:40:30
11	127083	p251806	4ba7c721133ef651ca54a03551746708	Ms.	CA	2016-11-14 22:57:28
12	19090	p051126	5e52c92b7e3c472aad247a239d345543	Mrs.	NY	2016-05-23 15:46:02
13	15126	p003874	178f6ae765cd4e0fb143a77c47fd65e2	Mrs.	OK	2016-10-17 09:49:27
14	62232	p233127	424819801de22a60bba7d0f4354d0258	Ms.	MA	2017-02-14 16:29:10
15	67303	p132832	bb6d6d054824fa01576ab38dfa2be160	Ms.	TX	2016-10-05 21:05:38
16	127215	p174627	4ad7e280fddff889e1355cc9f29c3b89	Mrs.	FL	2017-01-18 10:59:05
17	157771	p152491	e39abda057354c979c5b075cffbe5f88	Ms.	NV	2016-11-23 17:14:17
18	122186	p196421	fcd9b003fc1891383f340a89da02a1a6	Mrs.	GA	2016-08-28 15:04:42
19	146331	p058343	8e07a98deb1bc74c75b97521e05b1691	Ms.	OH	2016-08-06 13:05:20
20	75560	p052326	e0c1aad1f71badeff703fadc15f57680	Mrs.	PA	2016-10-07 18:27:02
21	132078	p187097	2d4a4d2d774e5c2fdd25b2ba0e7341f8	Mrs.	NC	2016-05-17 19:45:13

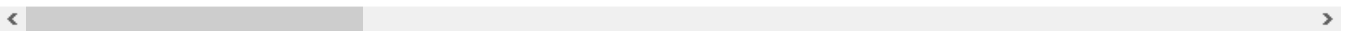
22	44610	med:0	p165540	id30f08fbe02eba5453c4ce2e857e88eb4	teacher_id	Ms.	teacher_prefix	CA	school_state	2016-09-01 10:09:15	project_submitted_da
23	8636		p219330	258ef2e6ab5ce007ac6764ce15d261ba		Mr.		AL		2017-01-10 11:41:06	
24	21478		p126524	74f8690562c44fc88f65f845b9fe61d0		Mrs.		FL		2017-03-31 12:34:44	
25	20142		p009037	b8bf3507cee960d5fedcb27719df2d59		Mrs.		AL		2017-03-09 15:36:20	
26	33903		p040091	7a0a5de5ed94e7036946b1ac3eaa99d0		Ms.		TX		2016-09-18 22:10:40	
27	1156		p161033	efdc3cf14d136473c9f62becc00d4cec		Teacher		LA		2016-11-06 16:02:31	
28	35430		p085706	22c8184c4660f1c589bea061d14b7f35		Mrs.		GA		2017-01-27 12:34:59	
29	22088		p032018	45f16a103f1e00b7439861d4e0728a59		Mrs.		VA		2016-07-15 12:58:40	
...	
109215	127181		p077978	91f5c69bf72c82edb9bc1f55596d8d95		Mrs.		IL		2017-01-10 14:08:28	
109216	65838		p042022	9a6784108c76576565f46446594f99c4		Teacher		FL		2016-07-26 22:43:52	
109217	21062		p064087	19c622a38a0cd76c2e9dbcc40541fabd		Mrs.		WI		2016-09-18 13:15:13	
109218	81490		p117254	031e299278ac511616b2950fc1312a55		Teacher		NY		2016-07-03 23:09:29	

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_da
109219	69138	p152194	6f6e951e435aa9dc966091945414bcc4	Ms.	NC	2016-12-01 20:29:04
109220	5110	p041136	6db62616b4ef6efc2310088f7ea0ae14	Ms.	GA	2017-02-15 14:07:07
109221	109630	p257774	651866d8215616f65934aafcbee21bf5	Ms.	NY	2016-05-23 20:36:51
109222	177841	p079425	c628dff071aa8028b08a5d4972bef2a1	Mrs.	NC	2016-11-14 21:04:43
109223	65359	p085810	1d286ff10ee3982b2b47813f1e415ef2	Ms.	CA	2016-08-12 09:19:22
109224	55643	p146149	e15cd063caa1ce11a45f2179535105f2	Mrs.	NY	2016-10-19 10:10:04
109225	103666	p191845	d0603199630760d8d0eb003108208998	Mrs.	LA	2016-10-14 18:05:17
109226	121219	p055363	523f95270c6aec82bee90e3931ceeeeca	Mrs.	CO	2016-09-06 23:19:17
109227	117282	p235512	ee59900af64d9244487e7ed87d0bc423	Ms.	NY	2016-08-09 21:06:33
109228	170085	p248898	9d7a4dae637d1a170778e2db1515e574	Mrs.	AZ	2016-09-17 09:58:59
109229	36083	p204774	c116af7435274872bea9ff123a69cf6a	Mrs.	MD	2017-03-14 19:59:52
109230	155847	p120664	b90258ab009b84e0dc11a7186d597141	Ms.	AZ	2016-12-21 16:36:26

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_da
109231	52918	p057638	dd68d9fbae85933c0173c13f66291cbe	Ms.	NY	2017-03-29 20:06:10
109232	69971	p105083	9636fcacbf65eb393133a94c83c4a0d4	Mrs.	TX	2017-01-07 14:50:08
109233	120581	p254202	2950019dd34581dbcdcaae683e74207a	Mrs.	OH	2016-08-14 08:27:24
109234	115336	p056813	07fd2c09f8dfcc74dbb161e1ec3df1fe	Mrs.	IN	2016-05-05 13:03:58
109235	32628	p143363	5b42211690ca8418c7c839436d0b7e49	Mrs.	WI	2016-08-01 21:17:33
109236	156548	p103958	8b9a9dc5bd4aa0301b0ff416e2ed29f6	Mrs.	MN	2016-08-15 17:01:00
109237	93971	p257729	58c112dcb2f1634a4d4236bf0dcdcb31	Mrs.	MD	2016-08-25 13:09:19
109238	36517	p180358	3e5c98480f4f39d465837b2955df6ae0	Mrs.	MD	2016-06-24 11:48:12
109239	34811	p080323	fe10e79b7aeb570dfac87eeea7e9a8f1	Mrs.	SC	2017-03-09 20:00:33
109240	38267	p048540	fadf72d6cd83ce6074f9be78a6fcd374	Mr.	MO	2016-06-17 12:02:31
109241	169142	p166281	1984d915cc8b91aa16b4d1e6e39296c6	Ms.	NJ	2017-01-11 12:49:39
109242	143653	p155633	cdbfd04aa041dc6739e9e576b1fb1478	Mrs.	NJ	2016-08-25 17:11:32

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_da
109243	164599	p206114	6d5675dbfafa1371f0e2f6f1b716fe2d	Mrs.	NY	2016-07-29 17:53:15
109244	128381	p191189	ca25d5573f2bd2660f7850a886395927	Ms.	VA	2016-06-29 09:17:01

109245 rows × 20 columns



2.2 Make Data Model Ready: encoding numerical, categorical features

In [210]:

```
#one hot encoding for School_State
```

```
vectorizer=CountVectorizer()
vectorizer.fit(x_train['school_state'].values)
```

```
school_state_features=vectorizer.get_feature_names()
```

```
x_train_state_oh = vectorizer.transform(x_train['school_state'].values)
x_cv_state_oh = vectorizer.transform(x_cv['school_state'].values)
x_test_state_oh = vectorizer.transform(x_test['school_state'].values)
```

```
print("After vectorizations")
print(x_train_state_oh.shape, y_train.shape)
print(x_cv_state_oh.shape, y_cv.shape)
print(x_test_state_oh.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

After vectorizations

```
(49039, 51) (49039,)
(24155, 51) (24155,)
(36051, 51) (36051,)
['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']
=====
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [211]:

```
#One hot encoding on Teacher_prefix
```

```
vectorizer = CountVectorizer()
vectorizer.fit(x_train['teacher_prefix'].values) # fit has to happen only on train data
```

```
# we use the fitted CountVectorizer to convert the text to vector
```

```
x_train_teacher_oh = vectorizer.transform(x_train['teacher_prefix'].values)
x_cv_teacher_oh = vectorizer.transform(x_cv['teacher_prefix'].values)
x_test_teacher_oh = vectorizer.transform(x_test['teacher_prefix'].values)
```

```
teacher_prefix_features=vectorizer.get_feature_names()
print("After vectorizations")
print(x_train_teacher_oh.shape, y_train.shape)
print(x_cv_teacher_oh.shape, y_cv.shape)
print(x_test_teacher_oh.shape, y_test.shape)
print(vectorizer.get feature names())
```

```
print("="*100)
```

After vectorizations

```
(49039, 5) (49039,)
(24155, 5) (24155,)
(36051, 5) (36051,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
```

In [212]:

```
#One hot encoding on project_grade_category
vocabulary=project_data['project_grade_category'].unique()
vectorizer = CountVectorizer(vocabulary=vocabulary,lowercase=False,binary=True)
vectorizer.fit(x_train['project_grade_category'].values) # fit has to happen only on train data

# used the fitted CountVectorizer to convert the text to vector

x_train_project_grade_category_ohe = vectorizer.transform(x_train['project_grade_category'].values)
x_cv_project_grade_category_ohe = vectorizer.transform(x_cv['project_grade_category'].values)
x_test_project_grade_category_ohe = vectorizer.transform(x_test['project_grade_category'].values)

project_grade_features=vectorizer.get_feature_names()

print("After vectorizations")
print(x_train_project_grade_category_ohe.shape, y_train.shape)
print(x_cv_project_grade_category_ohe.shape, y_cv.shape)
print(x_test_project_grade_category_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

After vectorizations

```
(49039, 4) (49039,)
(24155, 4) (24155,)
(49039, 4) (36051,)
['Grades PreK-2', 'Grades 6-8', 'Grades 3-5', 'Grades 9-12']
```

In [213]:

```
#One hot encoding on clean_categories

vocabulary=list(sorted_cat_dict.keys())
vectorizer = CountVectorizer(vocabulary=vocabulary,lowercase=False,binary=True)
vectorizer.fit(x_train['clean_categories'].values) # fit has to happen only on train data

# use the fitted CountVectorizer to convert the text to vector

x_train_clean_categories_ohe = vectorizer.transform(x_train['clean_categories'].values)
x_cv_clean_categories_ohe = vectorizer.transform(x_cv['clean_categories'].values)
x_test_clean_categories_ohe = vectorizer.transform(x_test['clean_categories'].values)

clean_categories_features=vectorizer.get_feature_names()

print("After vectorizations")
print(x_train_clean_categories_ohe.shape, y_train.shape)
print(x_cv_clean_categories_ohe.shape, y_cv.shape)
print(x_test_clean_categories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

After vectorizations

```
(49039, 9) (49039,)
(24155, 9) (24155,)
(36051, 9) (36051,)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
```

In [214]:

```
#One hot encoding on clean_subcategories
```

```

#one hot encoding on clean_subcategories
vocabulary=list(sorted_sub_cat_dict.keys())
vectorizer = CountVectorizer(vocabulary=vocabulary,lowercase=False,binary=True)
vectorizer.fit(x_train['clean_subcategories'].values) # fit has to happen only on train data

# use the fitted CountVectorizer to convert the text to vector

x_train_clean_subcategories_ohe = vectorizer.transform(x_train['clean_subcategories'].values)
x_cv_clean_subcategories_ohe = vectorizer.transform(x_cv['clean_subcategories'].values)
x_test_clean_subcategories_ohe = vectorizer.transform(x_test['clean_subcategories'].values)

clean_sub_features=vectorizer.get_feature_names()
print("After vectorizations")
print(x_train_clean_subcategories_ohe.shape, y_train.shape)
print(x_cv_clean_subcategories_ohe.shape, y_cv.shape)
print(x_test_clean_subcategories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)

```

```

After vectorizations
(49039, 30) (49039,)
(24155, 30) (24155,)
(36051, 30) (36051,)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_
Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'Perf
ormingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geogr
aphy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualA
rts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Liter
acy']
=====

```

In [215]:

```

# Column Standardization on Price
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(x_train['price'].values.reshape(-1,1))

x_train_price_stand=scaler.transform(x_train['price'].values.reshape(-1,1))
x_cv_price_stand=scaler.transform(x_cv['price'].values.reshape(-1,1))
x_test_price_stand=scaler.transform(x_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(x_train_price_stand.shape, y_train.shape)
print(x_cv_price_stand.shape, y_cv.shape)
print(x_test_price_stand.shape, y_test.shape)

```

```

After vectorizations
(49039, 1) (49039,)
(24155, 1) (24155,)
(36051, 1) (36051,)

```

In [216]:

```

#Performing column Standardization on Quantity

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train['quantity'].values.reshape(-1,1))

x_train_quan_stand=scaler.transform(x_train['quantity'].values.reshape(-1,1))
x_cv_quan_stand=scaler.transform(x_cv['quantity'].values.reshape(-1,1))
x_test_quan_stand=scaler.transform(x_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(x_train_quan_stand.shape, y_train.shape)
print(x_cv_quan_stand.shape, y_cv.shape)
print(x_test_quan_stand.shape, y_test.shape)
print((x_train_quan_stand))

```

```

After vectorizations
(49039, 1) (49039,)
(24155, 1) (24155,)
(36051, 1) (36051,)

```

```
(36051, 1) (36051,)
[[-0.54258655]
 [-0.46536001]
 [-0.19506713]
 ...
 [ 0.53858499]
 [-0.46536001]
 [-0.61981309]]
```

In [217]:

```
#Performing column Standardization on teacher_number_of_previously_posted_projects

from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

x_train_prev_subj_stand=scaler.transform(x_train['teacher_number_of_previously_posted_projects'].values
.reshape(-1,1))
x_cv_prev_subj_stand=scaler.transform(x_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
x_test_prev_subj_stand=scaler.transform(x_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print("After vectorizations")
print(x_train_prev_subj_stand.shape, y_train.shape)
print(x_cv_prev_subj_stand.shape, y_cv.shape)
print(x_test_prev_subj_stand.shape, y_test.shape)
```

```
After vectorizations
(49039, 1) (49039,)
(24155, 1) (24155,)
(36051, 1) (36051,)
```

In [228]:

```
#text Vectorization for Project title with BOW
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,3))
vectorizer.fit(x_train['title'].values)

x_train_title_bow = vectorizer.transform(x_train['title'].values)
x_cv_title_bow = vectorizer.transform(x_cv['title'].values)
x_test_title_bow = vectorizer.transform(x_test['title'].values)

print("After vectorizations")
print(x_train_title_bow.shape, y_train.shape)
print(x_cv_title_bow.shape, y_cv.shape)
print(x_test_title_bow.shape, y_test.shape)
print("=="*100)
```

```
After vectorizations
(49039, 4034) (49039,)
(24155, 4034) (24155,)
(36051, 4034) (36051,)
```

=====

In [229]:

```
#Text vectorization on preprocessed_essay with BOW

#BOW for project_title
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,2), max_features=5000)
vectorizer.fit(x_train['essay'].values)

x_train_essay_bow = vectorizer.transform(x_train['essay'].values)
x_cv_essay_bow = vectorizer.transform(x_cv['essay'].values)
x_test_essay_bow = vectorizer.transform(x_test['essay'].values)

print("After vectorizations")
print(x_train_essay_bow.shape, y_train.shape)
print(x_cv_essay_bow.shape, y_cv.shape)
```



```
print(x_test_essay_bow.shape, y_test.shape)
print("="*100)
```

After vectorizations
(49039, 5000) (49039,)
(24155, 5000) (24155,)
(36051, 5000) (36051,)

In [230]:

```
#TFIDF for essay
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,2),max_features=5000)
vectorizer.fit(x_train['essay'].values)

x_train_essay_tfidf = vectorizer.transform(x_train['essay'].values)
x_cv_essay_tfidf = vectorizer.transform(x_cv['essay'].values)
x_test_essay_tfidf = vectorizer.transform(x_test['essay'].values)

print("After vectorizations")
print(x_train_essay_tfidf.shape, y_train.shape)
print(x_cv_essay_tfidf.shape, y_cv.shape)
print(x_test_essay_tfidf.shape, y_test.shape)
print("="*100)
```

After vectorizations
(49039, 5000) (49039,)
(24155, 5000) (24155,)
(36051, 5000) (36051,)

In [231]:

```
#TFIDF on project_title
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,3))
vectorizer.fit(x_train['title'].values)

x_train_title_tfidf = vectorizer.transform(x_train['title'].values)
x_cv_title_tfidf = vectorizer.transform(x_cv['title'].values)
x_test_title_tfidf = vectorizer.transform(x_test['title'].values)

print("After vectorizations")
print(x_train_title_tfidf.shape, y_train.shape)
print(x_cv_title_tfidf.shape, y_cv.shape)
print(x_test_title_tfidf.shape, y_test.shape)
print("="*100)
```

After vectorizations
(49039, 4034) (49039,)
(24155, 4034) (24155,)
(36051, 4034) (36051,)

In [232]:

```
from scipy.sparse import hstack

x_tr_bow=hstack((x_train_title_bow,x_train_essay_bow,x_train_clean_subcategories_ohe,x_train_clean_categories_ohe,x_train_project_grade_category_ohe,x_train_teacher_ohe,x_train_state_ohe,x_train_prev_subj_stand,x_train_quan_stand,x_train_price_stand)).tocsr()
x_te_bow=hstack((x_test_title_bow,x_test_essay_bow,x_test_clean_subcategories_ohe,x_test_clean_categories_ohe,x_test_project_grade_category_ohe,x_test_teacher_ohe,x_test_state_ohe,x_test_prev_subj_stand,x_test_quan_stand,x_test_price_stand)).tocsr()
x_cv_bow=hstack((x_cv_title_bow,x_cv_essay_bow,x_cv_clean_subcategories_ohe,x_cv_clean_categories_ohe,x_cv_project_grade_category_ohe,x_cv_teacher_ohe,x_cv_state_ohe,x_cv_prev_subj_stand,x_cv_quan_stand,x_cv_price_stand)).tocsr()

print("Final Data matrix")
print(x_tr_bow.shape, y_train.shape)
```

```
print(x_cv_bow.shape, y_cv.shape)
print(x_te_bow.shape, y_test.shape)
print("="*100)
```

Final Data matrix

```
(49039, 9136) (49039,)
(24155, 9136) (24155,)
(36051, 9136) (36051,)
```

In [233]:

```
from scipy.sparse import hstack
```

```
x_tr_tfidf=hstack((x_train_title_tfidf,x_train_essay_tfidf,x_train_clean_subcategories_ohe,x_train_clean_categories_ohe,x_train_project_grade_category_ohe,x_train_teacher_ohe,x_train_state_ohe,x_train_prev_subj_stand,x_train_quan_stand,x_train_price_stand)).tocsr()
x_te_tfidf=hstack((x_test_title_tfidf,x_test_essay_tfidf,x_test_clean_subcategories_ohe,x_test_clean_categories_ohe,x_test_project_grade_category_ohe,x_test_teacher_ohe,x_test_state_ohe,x_test_prev_subj_stand,x_test_quan_stand,x_test_price_stand)).tocsr()
x_cv_tfidf=hstack((x_cv_title_tfidf,x_cv_essay_tfidf,x_cv_clean_subcategories_ohe,x_cv_clean_categories_ohe,x_cv_project_grade_category_ohe,x_cv_teacher_ohe,x_cv_state_ohe,x_cv_prev_subj_stand,x_cv_quan_stand,x_cv_price_stand)).tocsr()

print("Final Data matrix")
print(x_tr_tfidf.shape, y_train.shape)
print(x_cv_tfidf.shape, y_cv.shape)
print(x_te_tfidf.shape, y_test.shape)
print("="*100)
```

Final Data matrix

```
(49039, 9136) (49039,)
(24155, 9136) (24155,)
(36051, 9136) (36051,)
```

2.4 Applying Logistic Regression on different kind of featurization as mentioned in the instructions

Logistic Regression with BOW representation of text

In [234]:

```
#Finding the Best possible value of Hyperparameter(HyperParameter tuning)

from sklearn.metrics import roc_auc_score
train_auc=[]
cv_auc=[]
c_list=[10**(-4),10**(-3),10**(-2),10**(-1),10**(0),10**(1),10**(2),10**(3),10**(4)]
for i in c_list:
    clf=LogisticRegression(C=i,class_weight='balanced')
    clf.fit(x_tr_bow,y_train)
    y_train_pred=clf.predict_proba(x_tr_bow)[:,-1]
    y_cv_pred=clf.predict_proba(x_cv_bow)[:,-1]
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv,y_cv_pred))
```

In []:

```
plt.plot(np.log10(c_list), train_auc, label='Train AUC')
plt.plot(np.log10(c_list), cv_auc, label='CV AUC')

plt.scatter(np.log10(c_list), train_auc, label='Train AUC points')
plt.scatter(np.log10(c_list), cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
```

```
plt.ylabel("AUC")
plt.title("ERROR PLOTS (BOW)")
plt.grid()
plt.show()
```

In [237]:

```
best_c=10**-3

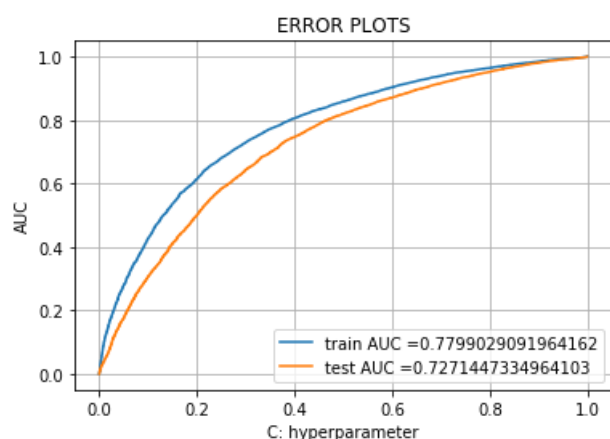
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression

clf_final=LogisticRegression(C=best_c, class_weight='balanced')
clf_final.fit(x_tr_bow, y_train)

y_train_pred=clf_final.predict_proba(x_tr_bow)[:,1]
y_test_pred=clf_final.predict_proba(x_te_bow)[:,1]

train_fpr_bow, train_tpr_bow, tr_thresholds_bow = roc_curve(y_train, y_train_pred)
test_fpr_bow, test_tpr_bow, te_thresholds_bow = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr_bow, train_tpr_bow, label="train AUC =" + str(auc(train_fpr_bow, train_tpr_bow)))
plt.plot(test_fpr_bow, test_tpr_bow, label="test AUC =" + str(auc(test_fpr_bow, test_tpr_bow)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [238]:

```
def predict(proba, threshold, fpr, tpr):

    t = threshold[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [239]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm_train=confusion_matrix(y_train, predict(y_train_pred, tr_thresholds_bow, train_fpr_bow, train_fpr_bow))
print("Test confusion matrix")
cm_test=confusion_matrix(y_test, predict(y_test_pred, tr_thresholds_bow, test_fpr_bow, test_fpr_bow))
```

Train confusion matrix
the maximum value of $\text{tpr} \cdot (1 - \text{fpr})$ 0.24999999546531537 for threshold 0.413
Test confusion matrix
the maximum value of $\text{tpr} \cdot (1 - \text{fpr})$ 0.2499999244983697 for threshold 0.461

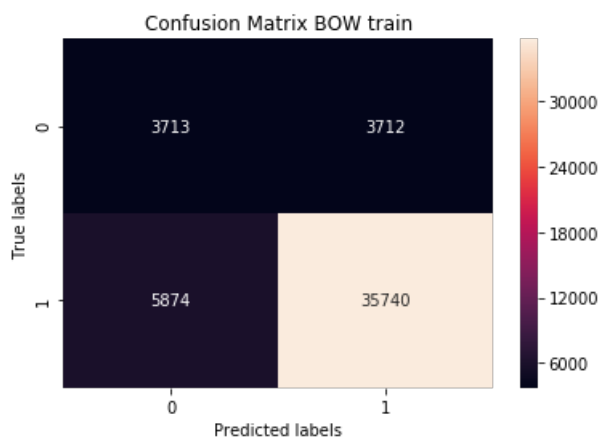
In [246]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(cm_train, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

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

```
1    41614
0     7425
Name: project_is_approved, dtype: int64
```



In [248]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(cm_test, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

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

```
1    30592
0     5459
Name: project_is_approved, dtype: int64
```



Logistic Regression with TFIDF representation of text

In [250]:

```
#Finding the Best possible value of Hyperparameter(HyperParameter tuning) (TFIDF)

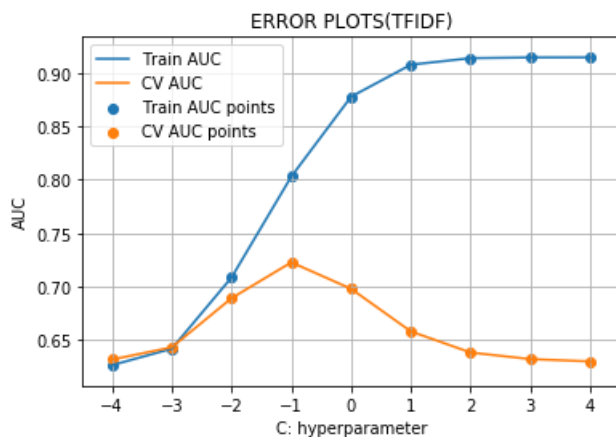
from sklearn.metrics import roc_auc_score
train_auc_tfidf=[]
cv_auc_tfidf=[]
c_list=[10**(-4),10**(-3),10**(-2),10**(-1),10**(0),10**(1),10**(2),10**(3),10**(4)]
for i in c_list:
    clf=LogisticRegression(C=i,class_weight='balanced')
    clf.fit(x_tr_tfidf,y_train)
    y_train_pred=clf.predict_proba(x_tr_tfidf)[:,-1]
    y_cv_pred=clf.predict_proba(x_cv_tfidf)[:,-1]
    train_auc_tfidf.append(roc_auc_score(y_train,y_train_pred))
    cv_auc_tfidf.append(roc_auc_score(y_cv,y_cv_pred))
```

In [252]:

```
plt.plot(np.log10(c_list), train_auc_tfidf, label='Train AUC')
plt.plot(np.log10(c_list), cv_auc_tfidf, label='CV AUC')

plt.scatter(np.log10(c_list), train_auc_tfidf, label='Train AUC points')
plt.scatter(np.log10(c_list), cv_auc_tfidf, label='CV AUC points')

plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS (TFIDF) ")
plt.grid()
plt.show()
```



In [253]:

```
best_c=0.1

from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression

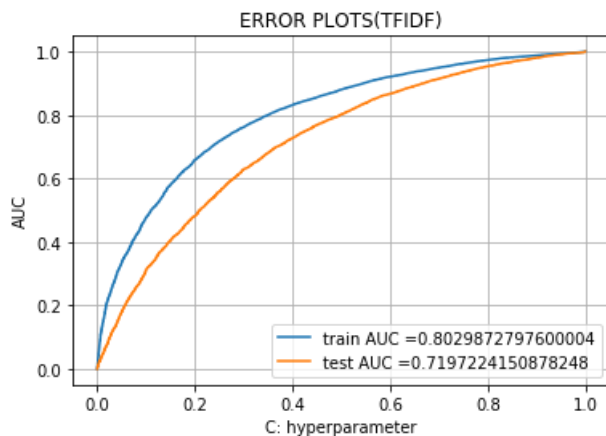
clf_tfidf=LogisticRegression(C=best_c,class_weight='balanced')
clf_tfidf.fit(x_tr_tfidf,y_train)

y_train_pred_tfidf=clf_tfidf.predict_proba(x_tr_tfidf)[:,-1]
y_test_pred_tfidf=clf_tfidf.predict_proba(x_te_tfidf)[:,-1]

train_fpr_tfidf, train_tpr_tfidf, tr_thresholds_tfidf = roc_curve(y_train, y_train_pred_tfidf)
test_fpr_tfidf, test_tpr_tfidf, te_thresholds_tfidf = roc_curve(y_test, y_test_pred_tfidf)

plt.plot(train_fpr_tfidf, train_tpr_tfidf, label="train AUC =" +str(auc(train_fpr_tfidf, train_tpr_tfidf)))
plt.plot(test_fpr_tfidf, test_tpr_tfidf, label="test AUC =" +str(auc(test_fpr_tfidf, test_tpr_tfidf)))
```

```
plt.plot(test_fpr_tfidf, test_tpr_tfidf, label="test AUC ="+str(auc(test_fpr_tfidf, test_tpr_tfidf)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS (TFIDF)")
plt.grid()
plt.show()
```



In [545]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm_train=confusion_matrix(y_train, predict(y_train_pred_tfidf, tr_thresholds_tfidf, train_fpr_tfidf, tr
ain_fpr_tfidf))
print("Test confusion matrix")
cm_test=confusion_matrix(y_test, predict(y_test_pred_tfidf, tr_thresholds_tfidf, test_fpr_tfidf, test_f
pr_tfidf))
```

Train confusion matrix

the maximum value of $tpr \cdot (1-fpr)$ 0.24999999546531537 for threshold 0.405

Test confusion matrix

the maximum value of $tpr \cdot (1-fpr)$ 0.24999999161092995 for threshold 0.452

In [546]:

```
import seaborn as sns
import matplotlib.pyplot as plt

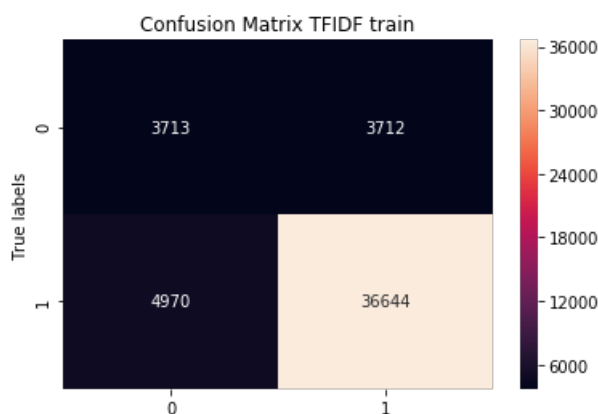
ax= plt.subplot()
sns.heatmap(cm_train, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

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

1 41614

0 7425

Name: project_is_approved, dtype: int64



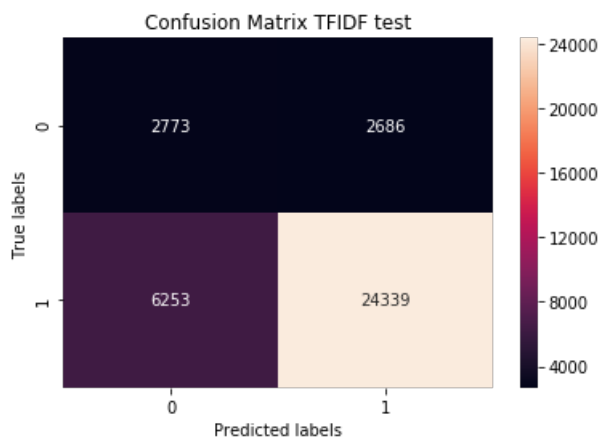
In [547]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(cm_test, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

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

```
1    30592
0     5459
Name: project_is_approved, dtype: int64
```



Logistic Regression with AverageWord2Vec representation of text

In [263]:

```
def avgw2vecessay(preprocessed_essays):

    from tqdm import tqdm
    avg_vectors_essay_train = []; # the avg-w2v for each sentence/review is stored in this list
    for sentence in tqdm(preprocessed_essays): # 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_vectors_essay_train.append(vector)

    return avg_vectors_essay_train
```

In [278]:

```
def avgw2vectitle(preprocessed_title):
    from tqdm import tqdm
    avg_vectors_title = []; # the avg-w2v for each sentence/review is stored in this list
    for sentence in (preprocessed_title): # 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:
```

```
if cnt_words != 0:
    vector /= cnt_words
    avg_vectors_title.append(vector)
return avg_vectors_title
```

In [274]:

```
avgw2vec_train_essays=avgw2vecessay(x_train['essay'].values)
avgw2vec_cv_essays=avgw2vecessay(x_cv['essay'].values)
avgw2vec_test_essays=avgw2vecessay(x_test['essay'].values)
```

```
0%|          | 0/49039
[00:00<?, ?it/s]
0%|          | 209/49039 [00:00<00:
23, 2074.80it/s]
1%|          | 413/49039 [00:00<00:
23, 2059.65it/s]
1%|          | 626/49039 [00:00<00:
23, 2075.84it/s]
2%|          | 842/49039 [00:00<00:
22, 2095.93it/s]
2%|          | 1053/49039 [00:00<00:
22, 2095.56it/s]
3%|          | 1271/49039 [00:00<00:
22, 2115.69it/s]
3%|          | 1471/49039 [00:00<00:
22, 2074.86it/s]
3%|          | 1674/49039 [00:00<00:
23, 2056.61it/s]
4%|          | 1888/49039 [00:00<00:
22, 2076.53it/s]
4%|          | 2088/49039 [00:01<00:
23, 1981.55it/s]
5%|          | 2282/49039 [00:01<00:
24, 1923.73it/s]
5%|          | 2472/49039 [00:01<00:
25, 1861.86it/s]
5%|          | 2677/49039 [00:01<00:
24, 1910.67it/s]
6%|          | 2868/49039 [00:01<00:
24, 1894.97it/s]
6%|          | 3057/49039 [00:01<00:
24, 1878.09it/s]
7%|          | 3245/49039 [00:01<00:
24, 1857.77it/s]
7%|          | 3431/49039 [00:01<00:
24, 1832.67it/s]
7%|          | 3635/49039 [00:01<00:
24, 1886.48it/s]
8%|          | 3826/49039 [00:01<00:
23, 1889.21it/s]
8%|          | 4016/49039 [00:02<00:
23, 1888.32it/s]
9%|          | 4206/49039 [00:02<00:
23, 1882.24it/s]
9%|          | 4412/49039 [00:02<00:
23, 1928.29it/s]
9%|          | 4606/49039 [00:02<00:
23, 1927.58it/s]
10%|         | 4800/49039 [00:02<00:
23, 1915.71it/s]
10%|         | 5002/49039 [00:02<00:
22, 1941.74it/s]
11%|         | 5198/49039 [00:02<00:
22, 1942.96it/s]
11%|         | 5396/49039 [00:02<00:
22, 1949.72it/s]
11%|         | 5602/49039 [00:02<00:
21, 1977.36it/s]
12%|         | 5800/49039 [00:02<00:
22, 1944.84it/s]
12%|         | 5995/49039 [00:03<00:
22, 1919.23it/s]
13%|         | 6188/49039 [00:03<00:
22, 1890.18it/s]
13%|         | 6378/49039 [00:03<00:
```









































22, 1877.82it/s]	6576/49039	[00:03<00:
13%	6566/49039	[00:03<00:
22, 1857.74it/s]		
14%	6774/49039	[00:03<00:
22, 1915.39it/s]		
14%	6986/49039	[00:03<00:
21, 1968.49it/s]		
15%	7184/49039	[00:03<00:
21, 1933.16it/s]		
15%	7379/49039	[00:03<00:
21, 1928.23it/s]		
15%	7586/49039	[00:03<00:
21, 1964.44it/s]		
16%	7783/49039	[00:04<00:
23, 1776.68it/s]		
16%	7983/49039	[00:04<00:
22, 1834.53it/s]		
17%	8197/49039	[00:04<00:
21, 1912.88it/s]		
17%	8392/49039	[00:04<00:
22, 1817.82it/s]		
17%	8577/49039	[00:04<00:
22, 1781.54it/s]		
18%	8758/49039	[00:04<00:
22, 1760.12it/s]		
18%	8975/49039	[00:04<00:
21, 1862.34it/s]		
19%	9167/49039	[00:04<00:
21, 1875.12it/s]		
19%	9357/49039	[00:04<00:
21, 1878.58it/s]		
19%	9553/49039	[00:04<00:
20, 1898.25it/s]		
20%	9752/49039	[00:05<00:
20, 1920.63it/s]		
20%	9945/49039	[00:05<00:
20, 1874.79it/s]		
21%	10136/49039	[00:05<00:
20, 1881.14it/s]		
21%	10325/49039	[00:05<00:
21, 1804.35it/s]		
21%	10510/49039	[00:05<00:
21, 1814.06it/s]		
22%	10693/49039	[00:05<00:
21, 1777.92it/s]		
22%	10872/49039	[00:05<00:
23, 1654.73it/s]		
23%	11040/49039	[00:05<00:
23, 1639.28it/s]		
23%	11222/49039	[00:05<00:
22, 1686.18it/s]		
23%	11421/49039	[00:06<00:
21, 1763.68it/s]		
24%	11624/49039	[00:06<00:
20, 1832.30it/s]		
24%	11823/49039	[00:06<00:
19, 1873.06it/s]		
25%	12033/49039	[00:06<00:
19, 1931.92it/s]		
25%	12228/49039	[00:06<00:
19, 1871.87it/s]		
25%	12419/49039	[00:06<00:
19, 1879.24it/s]		
26%	12633/49039	[00:06<00:
18, 1946.65it/s]		
26%	12830/49039	[00:06<00:
18, 1909.79it/s]		
27%	13049/49039	[00:06<00:
18, 1981.93it/s]		
27%	13254/49039	[00:06<00:
17, 1997.60it/s]		
27%	13460/49039	[00:07<00:
17, 2011.77it/s]		
28%	13662/49039	[00:07<00:
21, 1671.48it/s]		
28%	13845/49039	[00:07<00:
20, 1712.54it/s]		

20, 1712.94it/s]		
29%		14024/49039 [00:07<00:
22, 1584.64it/s]		
29%		14210/49039 [00:07<00:
21, 1655.06it/s]		
29%		14382/49039 [00:07<00:
20, 1670.46it/s]		
30%		14554/49039 [00:07<00:
20, 1661.97it/s]		
30%		14737/49039 [00:07<00:
20, 1705.54it/s]		
30%		14942/49039 [00:07<00:
19, 1792.64it/s]		
31%		15125/49039 [00:08<00:
19, 1753.37it/s]		
31%		15306/49039 [00:08<00:
19, 1766.19it/s]		
32%		15485/49039 [00:08<00:
19, 1753.73it/s]		
32%		15679/49039 [00:08<00:
18, 1802.20it/s]		
32%		15861/49039 [00:08<00:
18, 1787.68it/s]		
33%		16044/49039 [00:08<00:
18, 1796.29it/s]		
33%		16226/49039 [00:08<00:
18, 1799.45it/s]		
33%		16410/49039 [00:08<00:
18, 1807.51it/s]		
34%		16605/49039 [00:08<00:
17, 1844.06it/s]		
34%		16790/49039 [00:08<00:
17, 1799.07it/s]		
35%		16983/49039 [00:09<00:
17, 1832.63it/s]		
35%		17170/49039 [00:09<00:
17, 1839.72it/s]		
35%		17365/49039 [00:09<00:
16, 1867.53it/s]		
36%		17553/49039 [00:09<00:
17, 1782.43it/s]		
36%		17733/49039 [00:09<00:
17, 1773.27it/s]		
37%		17919/49039 [00:09<00:
17, 1794.64it/s]		
37%		18100/49039 [00:09<00:
17, 1795.31it/s]		
37%		18282/49039 [00:09<00:
17, 1798.74it/s]		
38%		18463/49039 [00:09<00:
17, 1726.19it/s]		
38%		18637/49039 [00:10<00:
17, 1726.70it/s]		
38%		18811/49039 [00:10<00:
17, 1716.70it/s]		
39%		18995/49039 [00:10<00:
17, 1748.14it/s]		
39%		19171/49039 [00:10<00:
17, 1742.83it/s]		
39%		19346/49039 [00:10<00:
17, 1725.60it/s]		
40%		19525/49039 [00:10<00:
16, 1740.71it/s]		
40%		19700/49039 [00:10<00:
17, 1632.98it/s]		
41%		19869/49039 [00:10<00:
17, 1646.16it/s]		
41%		20035/49039 [00:10<00:
17, 1632.16it/s]		
41%		20199/49039 [00:10<00:
18, 1574.71it/s]		
42%		20367/49039 [00:11<00:
17, 1601.53it/s]		
42%		20559/49039 [00:11<00:
16, 1682.04it/s]		
42%		20764/49039 [00:11<00:
15, 1774.53it/s]		
42%		20971/49039 [00:11<00:

[illegible]

10, 1959.19it/s]	58%		28282/49039 [00:15<00:
10, 1958.15it/s]	58%		28479/49039 [00:15<00:
10, 1946.00it/s]	58%		28676/49039 [00:15<00:
10, 1948.90it/s]	59%		28877/49039 [00:15<00:
10, 1962.65it/s]	59%		29078/49039 [00:15<00:
10, 1972.36it/s]	60%		29276/49039 [00:16<00:
10, 1970.34it/s]	60%		29491/49039 [00:16<00:
09, 2016.68it/s]	61%		29709/49039 [00:16<00:
09, 2058.79it/s]	61%		29921/49039 [00:16<00:
09, 2072.32it/s]	61%		30129/49039 [00:16<00:
09, 1987.25it/s]	62%		30329/49039 [00:16<00:
09, 1969.17it/s]	62%		30527/49039 [00:16<00:
09, 1916.95it/s]	63%		30731/49039 [00:16<00:
09, 1948.19it/s]	63%		30939/49039 [00:16<00:
09, 1981.80it/s]	63%		31138/49039 [00:16<00:
09, 1979.93it/s]	64%		31369/49039 [00:17<00:
08, 2064.55it/s]	64%		31577/49039 [00:17<00:
08, 1999.35it/s]	65%		31788/49039 [00:17<00:
08, 2026.89it/s]	65%		31999/49039 [00:17<00:
08, 2046.89it/s]	66%		32205/49039 [00:17<00:
08, 2028.11it/s]	66%		32425/49039 [00:17<00:
08, 2072.66it/s]	67%		32665/49039 [00:17<00:
07, 2156.67it/s]	67%		32882/49039 [00:17<00:
08, 2018.14it/s]	67%		33087/49039 [00:17<00:
08, 1915.72it/s]	68%		33282/49039 [00:18<00:
08, 1866.53it/s]	68%		33472/49039 [00:18<00:
08, 1798.17it/s]	69%		33682/49039 [00:18<00:
08, 1875.65it/s]	69%		33872/49039 [00:18<00:
08, 1873.27it/s]	69%		34062/49039 [00:18<00:
08, 1849.80it/s]	70%		34271/49039 [00:18<00:
07, 1912.01it/s]	70%		34464/49039 [00:18<00:
07, 1896.33it/s]	71%		34660/49039 [00:18<00:
07, 1910.89it/s]	71%		34852/49039 [00:18<00:
07, 1903.81it/s]	72%		35075/49039 [00:18<00:
07, 1987.28it/s]	72%		35276/49039 [00:19<00:
07, 1802.83it/s]	72%		35470/49039 [00:19<00:
07, 1838.08it/s]	73%		35687/49039 [00:19<00:
06, 1922.60it/s]	73%		35896/49039 [00:19<00:
06, 1966.03it/s]	74%		36006/49039 [00:19<00:







































74%		36096/49039	[00:19<00:
06, 1861.98it/s]			
74%		36293/49039	[00:19<00:
06, 1889.13it/s]			
74%		36485/49039	[00:19<00:
07, 1667.62it/s]			
75%		36691/49039	[00:19<00:
06, 1765.48it/s]			
75%		36889/49039	[00:19<00:
06, 1821.11it/s]			
76%		37076/49039	[00:20<00:
06, 1789.62it/s]			
76%		37266/49039	[00:20<00:
06, 1817.38it/s]			
76%		37451/49039	[00:20<00:
06, 1796.77it/s]			
77%		37664/49039	[00:20<00:
06, 1881.47it/s]			
77%		37855/49039	[00:20<00:
05, 1886.00it/s]			
78%		38046/49039	[00:20<00:
05, 1889.03it/s]			
78%		38237/49039	[00:20<00:
05, 1869.03it/s]			
78%		38458/49039	[00:20<00:
05, 1955.92it/s]			
79%		38656/49039	[00:20<00:
05, 1913.52it/s]			
79%		38870/49039	[00:20<00:
05, 1972.27it/s]			
80%		39089/49039	[00:21<00:
04, 2028.78it/s]			
80%		39294/49039	[00:21<00:
04, 1972.24it/s]			
81%		39495/49039	[00:21<00:
04, 1979.12it/s]			
81%		39694/49039	[00:21<00:
04, 1960.56it/s]			
81%		39891/49039	[00:21<00:
04, 1919.14it/s]			
82%		40084/49039	[00:21<00:
04, 1820.74it/s]			
82%		40286/49039	[00:21<00:
04, 1872.31it/s]			
83%		40475/49039	[00:21<00:
04, 1862.59it/s]			
83%		40698/49039	[00:21<00:
04, 1955.69it/s]			
83%		40898/49039	[00:22<00:
04, 1964.53it/s]			
84%		41098/49039	[00:22<00:
04, 1970.77it/s]			
84%		41297/49039	[00:22<00:
03, 1943.40it/s]			
85%		41499/49039	[00:22<00:
03, 1961.42it/s]			
85%		41696/49039	[00:22<00:
03, 1959.87it/s]			
85%		41895/49039	[00:22<00:
03, 1964.55it/s]			
86%		42092/49039	[00:22<00:
03, 1944.52it/s]			
86%		42297/49039	[00:22<00:
03, 1970.83it/s]			
87%		42502/49039	[00:22<00:
03, 1989.68it/s]			
87%		42703/49039	[00:22<00:
03, 1991.42it/s]			
87%		42903/49039	[00:23<00:
03, 1872.59it/s]			
88%		43106/49039	[00:23<00:
03, 1913.20it/s]			
88%		43310/49039	[00:23<00:
02, 1945.49it/s]			
89%		43506/49039	[00:23<00:
02, 1933.94it/s]			
89%		43708/49039	[00:23<00:

02, 1954.83it/s]	90% 	43905/49039 [00:23<00:
02, 1926.64it/s]	90% 	44103/49039 [00:23<00:
02, 1938.02it/s]	90% 	44298/49039 [00:23<00:
02, 1909.15it/s]	91% 	44495/49039 [00:23<00:
02, 1922.89it/s]	91% 	44688/49039 [00:23<00:
02, 1903.80it/s]	92% 	44879/49039 [00:24<00:
02, 1879.12it/s]	92% 	45086/49039 [00:24<00:
02, 1928.63it/s]	92% 	45312/49039 [00:24<00:
01, 2013.30it/s]	93% 	45515/49039 [00:24<00:
01, 1955.98it/s]	93% 	45712/49039 [00:24<00:
01, 1955.90it/s]	94% 	45909/49039 [00:24<00:
01, 1944.29it/s]	94% 	46124/49039 [00:24<00:
01, 1997.52it/s]	94% 	46325/49039 [00:24<00:
01, 1911.81it/s]	95% 	46525/49039 [00:24<00:
01, 1933.18it/s]	95% 	46720/49039 [00:25<00:
01, 1873.00it/s]	96% 	46925/49039 [00:25<00:
01, 1918.85it/s]	96% 	47118/49039 [00:25<00:
01, 1884.22it/s]	96% 	47311/49039 [00:25<00:
00, 1893.81it/s]	97% 	47502/49039 [00:25<00:
00, 1861.21it/s]	97% 	47701/49039 [00:25<00:
00, 1894.26it/s]	98% 	47909/49039 [00:25<00:
00, 1942.34it/s]	98% 	48104/49039 [00:25<00:
00, 1906.26it/s]	98% 	48296/49039 [00:25<00:
00, 1895.11it/s]	99% 	48486/49039 [00:25<00:
00, 1853.76it/s]	99% 	48672/49039 [00:26<00:
00, 1797.88it/s]	100% 	48860/49039 [00:26<00:
00, 1817.89it/s]	100% 	49039/49039 [00:26<00:
00, 1866.87it/s]	0% 	0/24155
[00:00<?, ?it/s]	1% 	199/24155 [00:00<00:
12, 1975.53it/s]	1% 	358/24155 [00:00<00:
12, 1836.91it/s]	2% 	534/24155 [00:00<00:
13, 1809.05it/s]	3% 	741/24155 [00:00<00:
12, 1876.29it/s]	4% 	927/24155 [00:00<00:
12, 1867.40it/s]	5% 	1151/24155 [00:00<00:
11, 1961.71it/s]	5% 	1328/24155 [00:00<00:
12, 1895.52it/s]	6% 	1523/24155 [00:00<00:
11, 1907.29it/s]	7% 	1710/24155 [00:00<00:
11, 1891.90it/s]	8% 	1894/24155 [00:01<00:
11, 1871.83it/s]	8% 	1894/24155 [00:01<00:

9% ██████████	12, 1764.02it/s]	2077/24155 [00:01<00:
9% ██████████	12, 1724.84it/s]	2252/24155 [00:01<00:
10% ██████████	12, 1759.63it/s]	2438/24155 [00:01<00:
11% ██████████	12, 1758.89it/s]	2615/24155 [00:01<00:
12% ██████████	13, 1638.06it/s]	2791/24155 [00:01<00:
12% ██████████	12, 1744.22it/s]	2998/24155 [00:01<00:
13% ██████████	11, 1822.37it/s]	3203/24155 [00:01<00:
14% ██████████	11, 1737.36it/s]	3389/24155 [00:01<00:
15% ██████████	11, 1787.51it/s]	3582/24155 [00:01<00:
16% ██████████	11, 1837.46it/s]	3780/24155 [00:02<00:
16% ██████████	11, 1829.34it/s]	3966/24155 [00:02<00:
17% ██████████	12, 1650.61it/s]	4151/24155 [00:02<00:
18% ██████████	11, 1667.28it/s]	4323/24155 [00:02<00:
19% ██████████	11, 1746.82it/s]	4521/24155 [00:02<00:
20% ██████████	10, 1829.10it/s]	4728/24155 [00:02<00:
20% ██████████	10, 1854.33it/s]	4921/24155 [00:02<00:
21% ██████████	10, 1830.72it/s]	5109/24155 [00:02<00:
22% ██████████	09, 1890.26it/s]	5315/24155 [00:02<00:
23% ██████████	09, 1903.74it/s]	5510/24155 [00:03<00:
24% ██████████	09, 1898.80it/s]	5702/24155 [00:03<00:
24% ██████████	09, 1827.37it/s]	5893/24155 [00:03<00:
25% ██████████	09, 1904.99it/s]	6106/24155 [00:03<00:
26% ██████████	09, 1848.15it/s]	6299/24155 [00:03<00:
27% ██████████	09, 1782.03it/s]	6486/24155 [00:03<00:
28% ██████████	09, 1886.05it/s]	6706/24155 [00:03<00:
29% ██████████	09, 1864.67it/s]	6898/24155 [00:03<00:
29% ██████████	09, 1857.17it/s]	7087/24155 [00:03<00:
30% ██████████	08, 1939.71it/s]	7305/24155 [00:03<00:
31% ██████████	08, 1983.75it/s]	7516/24155 [00:04<00:
32% ██████████	08, 1894.30it/s]	7716/24155 [00:04<00:
33% ██████████	08, 1886.65it/s]	7908/24155 [00:04<00:
34% ██████████	08, 1920.76it/s]	8110/24155 [00:04<00:
34% ██████████	08, 1928.20it/s]	8306/24155 [00:04<00:
35% ██████████	08, 1839.87it/s]	8500/24155 [00:04<00:
36% ██████████	08, 1865.01it/s]	8694/24155 [00:04<00:
37% ██████████	08, 1894.10it/s]	8892/24155 [00:04<00:
38% ██████████	07, 1950.14it/s]	9103/24155 [00:04<00:
39% ██████████	07, 1895.43it/s]	9300/24155 [00:05<00:
39% ██████████		9491/24155 [00:05<00:

08, 1809.83it/s]	40%		9686/24155 [00:05<00:
07, 1845.75it/s]	41%		9872/24155 [00:05<00:
07, 1819.11it/s]	42%		10080/24155 [00:05<00:
07, 1886.46it/s]	43%		10270/24155 [00:05<00:
07, 1864.25it/s]	43%		10458/24155 [00:05<00:
07, 1848.42it/s]	44%		10650/24155 [00:05<00:
07, 1865.35it/s]	45%		10838/24155 [00:05<00:
07, 1796.32it/s]	46%		11019/24155 [00:05<00:
07, 1765.04it/s]	46%		11200/24155 [00:06<00:
07, 1774.47it/s]	47%		11413/24155 [00:06<00:
06, 1864.43it/s]	48%		11607/24155 [00:06<00:
06, 1882.46it/s]	49%		11797/24155 [00:06<00:
06, 1829.32it/s]	50%		11982/24155 [00:06<00:
07, 1729.05it/s]	50%		12157/24155 [00:06<00:
06, 1721.34it/s]	51%		12341/24155 [00:06<00:
06, 1751.50it/s]	52%		12529/24155 [00:06<00:
06, 1784.57it/s]	53%		12729/24155 [00:06<00:
06, 1840.46it/s]	53%		12922/24155 [00:07<00:
06, 1862.34it/s]	54%		13128/24155 [00:07<00:
05, 1913.79it/s]	55%		13332/24155 [00:07<00:
05, 1945.72it/s]	56%		13550/24155 [00:07<00:
05, 2006.65it/s]	57%		13752/24155 [00:07<00:
05, 1937.19it/s]	58%		13947/24155 [00:07<00:
05, 1864.73it/s]	59%		14135/24155 [00:07<00:
05, 1775.50it/s]	59%		14315/24155 [00:07<00:
05, 1763.26it/s]	60%		14524/24155 [00:07<00:
05, 1846.46it/s]	61%		14722/24155 [00:07<00:
05, 1880.80it/s]	62%		14912/24155 [00:08<00:
05, 1833.52it/s]	63%		15117/24155 [00:08<00:
04, 1889.68it/s]	63%		15308/24155 [00:08<00:
04, 1874.96it/s]	64%		15515/24155 [00:08<00:
04, 1925.56it/s]	65%		15709/24155 [00:08<00:
04, 1807.42it/s]	66%		15906/24155 [00:08<00:
04, 1849.47it/s]	67%		16111/24155 [00:08<00:
04, 1901.68it/s]	67%		16303/24155 [00:08<00:
04, 1816.80it/s]	68%		16496/24155 [00:08<00:
04, 1845.32it/s]	69%		16682/24155 [00:09<00:
04, 1845.83it/s]	70%		16887/24155 [00:09<00:
03, 1898.67it/s]			

71%		17078/24155 [00:09<00:
03, 1811.86it/s]		
71%		17261/24155 [00:09<00:
03, 1761.12it/s]		
72%		17439/24155 [00:09<00:
03, 1747.38it/s]		
73%		17615/24155 [00:09<00:
03, 1747.33it/s]		
74%		17791/24155 [00:09<00:
03, 1736.98it/s]		
74%		17969/24155 [00:09<00:
03, 1745.91it/s]		
75%		18169/24155 [00:09<00:
03, 1811.34it/s]		
76%		18352/24155 [00:09<00:
03, 1802.43it/s]		
77%		18552/24155 [00:10<00:
03, 1853.70it/s]		
78%		18739/24155 [00:10<00:
03, 1765.32it/s]		
78%		18928/24155 [00:10<00:
02, 1797.22it/s]		
79%		19116/24155 [00:10<00:
02, 1816.83it/s]		
80%		19320/24155 [00:10<00:
02, 1875.27it/s]		
81%		19517/24155 [00:10<00:
02, 1898.85it/s]		
82%		19711/24155 [00:10<00:
02, 1906.90it/s]		
82%		19927/24155 [00:10<00:
02, 1972.42it/s]		
83%		20143/24155 [00:10<00:
01, 2020.87it/s]		
84%		20375/24155 [00:10<00:
01, 2098.15it/s]		
85%		20587/24155 [00:11<00:
01, 2057.39it/s]		
86%		20794/24155 [00:11<00:
01, 1980.02it/s]		
87%		20994/24155 [00:11<00:
01, 1913.74it/s]		
88%		21187/24155 [00:11<00:
01, 1812.23it/s]		
89%		21390/24155 [00:11<00:
01, 1868.57it/s]		
89%		21579/24155 [00:11<00:
01, 1833.03it/s]		
90%		21791/24155 [00:11<00:
01, 1906.85it/s]		
91%		21994/24155 [00:11<00:
01, 1937.98it/s]		
92%		22226/24155 [00:11<00:
00, 2034.91it/s]		
93%		22432/24155 [00:12<00:
00, 1934.87it/s]		
94%		22629/24155 [00:12<00:
00, 1907.33it/s]		
94%		22822/24155 [00:12<00:
00, 1887.41it/s]		
95%		23013/24155 [00:12<00:
00, 1890.03it/s]		
96%		23203/24155 [00:12<00:
00, 1872.33it/s]		
97%		23409/24155 [00:12<00:
00, 1921.00it/s]		
98%		23602/24155 [00:12<00:
00, 1908.14it/s]		
99%		23794/24155 [00:12<00:
00, 1831.30it/s]		
99%		23979/24155 [00:12<00:
00, 1775.01it/s]		
100%		24155/24155 [00:13<00:
00, 1854.63it/s]		
0%		0/36051
[00:00<?, ?it/s]		
1%		184/36051 [00:00<00:

19, 1826.66it/s]	1% 	401/36051 [00:00<00:
18, 1913.85it/s]	2% 	615/36051 [00:00<00:
17, 1972.66it/s]	2% 	801/36051 [00:00<00:
18, 1933.04it/s]	3% 	989/36051 [00:00<00:
18, 1912.55it/s]	3% 	1175/36051 [00:00<00:
18, 1892.24it/s]	4% 	1372/36051 [00:00<00:
18, 1910.85it/s]	4% 	1556/36051 [00:00<00:
18, 1884.77it/s]	5% 	1735/36051 [00:00<00:
18, 1822.90it/s]	5% 	1932/36051 [00:01<00:
18, 1860.83it/s]	6% 	2153/36051 [00:01<00:
17, 1949.64it/s]	7% 	2346/36051 [00:01<00:
17, 1916.38it/s]	7% 	2537/36051 [00:01<00:
17, 1871.07it/s]	8% 	2727/36051 [00:01<00:
17, 1875.59it/s]	8% 	2915/36051 [00:01<00:
17, 1850.73it/s]	9% 	3100/36051 [00:01<00:
18, 1819.30it/s]	9% 	3289/36051 [00:01<00:
17, 1836.04it/s]	10% 	3473/36051 [00:01<00:
17, 1827.77it/s]	10% 	3656/36051 [00:01<00:
18, 1746.30it/s]	11% 	3863/36051 [00:02<00:
17, 1828.70it/s]	11% 	4060/36051 [00:02<00:
17, 1865.05it/s]	12% 	4262/36051 [00:02<00:
16, 1905.03it/s]	12% 	4454/36051 [00:02<00:
17, 1768.80it/s]	13% 	4643/36051 [00:02<00:
17, 1799.73it/s]	13% 	4849/36051 [00:02<00:
16, 1866.93it/s]	14% 	5038/36051 [00:02<00:
16, 1847.84it/s]	14% 	5225/36051 [00:02<00:
16, 1839.51it/s]	15% 	5413/36051 [00:02<00:
16, 1847.50it/s]	16% 	5629/36051 [00:02<00:
15, 1927.38it/s]	16% 	5824/36051 [00:03<00:
15, 1896.40it/s]	17% 	6041/36051 [00:03<00:
15, 1967.04it/s]	17% 	6240/36051 [00:03<00:
15, 1969.59it/s]	18% 	6438/36051 [00:03<00:
15, 1956.76it/s]	18% 	6635/36051 [00:03<00:
15, 1910.86it/s]	19% 	6858/36051 [00:03<00:
14, 1992.68it/s]	20% 	7059/36051 [00:03<00:
14, 1941.64it/s]	20% 	7291/36051 [00:03<00:
14, 2037.61it/s]	21% 	7497/36051 [00:03<00:
14, 1969.79it/s]	21% 	7696/36051 [00:04<00:
15, 1871.65it/s]		

22%		7886/36051	[00:04<00:
15,	1827.56it/s]		
22%		8109/36051	[00:04<00:
14,	1928.51it/s]		
23%		8312/36051	[00:04<00:
14,	1953.75it/s]		
24%		8510/36051	[00:04<00:
14,	1951.52it/s]		
24%		8707/36051	[00:04<00:
14,	1913.06it/s]		
25%		8924/36051	[00:04<00:
13,	1979.55it/s]		
25%		9124/36051	[00:04<00:
13,	1969.66it/s]		
26%		9324/36051	[00:04<00:
13,	1974.37it/s]		
26%		9523/36051	[00:04<00:
13,	1957.31it/s]		
27%		9720/36051	[00:05<00:
13,	1933.84it/s]		
27%		9914/36051	[00:05<00:
13,	1920.01it/s]		
28%		10107/36051	[00:05<00:
13,	1901.84it/s]		
29%		10298/36051	[00:05<00:
13,	1894.49it/s]		
29%		10497/36051	[00:05<00:
13,	1918.12it/s]		
30%		10690/36051	[00:05<00:
14,	1799.77it/s]		
30%		10919/36051	[00:05<00:
13,	1919.76it/s]		
31%		11115/36051	[00:05<00:
13,	1910.62it/s]		
31%		11309/36051	[00:05<00:
13,	1849.64it/s]		
32%		11520/36051	[00:06<00:
12,	1916.91it/s]		
32%		11714/36051	[00:06<00:
12,	1919.62it/s]		
33%		11928/36051	[00:06<00:
12,	1976.65it/s]		
34%		12128/36051	[00:06<00:
12,	1917.02it/s]		
34%		12322/36051	[00:06<00:
12,	1875.25it/s]		
35%		12511/36051	[00:06<00:
12,	1870.02it/s]		
35%		12727/36051	[00:06<00:
11,	1944.50it/s]		
36%		12923/36051	[00:06<00:
12,	1899.93it/s]		
36%		13153/36051	[00:06<00:
11,	2000.70it/s]		
37%		13356/36051	[00:06<00:
11,	1975.87it/s]		
38%		13556/36051	[00:07<00:
11,	1955.58it/s]		
38%		13753/36051	[00:07<00:
11,	1904.70it/s]		
39%		13945/36051	[00:07<00:
11,	1882.76it/s]		
39%		14135/36051	[00:07<00:
11,	1883.79it/s]		
40%		14324/36051	[00:07<00:
11,	1853.93it/s]		
40%		14540/36051	[00:07<00:
11,	1932.45it/s]		
41%		14735/36051	[00:07<00:
11,	1933.47it/s]		
41%		14960/36051	[00:07<00:
10,	2014.70it/s]		
42%		15163/36051	[00:07<00:
10,	2008.74it/s]		
43%		15370/36051	[00:08<00:
10,	2022.57it/s]		
43%		15580/36051	[00:08<00:

10, 2040.82it/s]	44%		15785/36051 [00:08<00:
10, 1963.16it/s]	44%		15983/36051 [00:08<00:
10, 1935.02it/s]	45%		16178/36051 [00:08<00:
10, 1879.63it/s]	45%		16374/36051 [00:08<00:
10, 1898.99it/s]	46%		16565/36051 [00:08<00:
11, 1752.25it/s]	46%		16743/36051 [00:08<00:
11, 1716.14it/s]	47%		16917/36051 [00:08<00:
11, 1651.15it/s]	47%		17085/36051 [00:08<00:
11, 1632.05it/s]	48%		17292/36051 [00:09<00:
10, 1739.45it/s]	48%		17469/36051 [00:09<00:
10, 1714.25it/s]	49%		17652/36051 [00:09<00:
10, 1743.75it/s]	50%		17868/36051 [00:09<00:
09, 1847.28it/s]	50%		18056/36051 [00:09<00:
10, 1764.64it/s]	51%		18241/36051 [00:09<00:
09, 1785.63it/s]	51%		18422/36051 [00:09<00:
10, 1752.70it/s]	52%		18615/36051 [00:09<00:
09, 1797.39it/s]	52%		18835/36051 [00:09<00:
09, 1898.20it/s]	53%		19028/36051 [00:10<00:
09, 1870.39it/s]	53%		19234/36051 [00:10<00:
08, 1919.59it/s]	54%		19428/36051 [00:10<00:
08, 1904.55it/s]	54%		19620/36051 [00:10<00:
08, 1877.15it/s]	55%		19849/36051 [00:10<00:
08, 1980.72it/s]	56%		20050/36051 [00:10<00:
08, 1906.25it/s]	56%		20257/36051 [00:10<00:
08, 1948.07it/s]	57%		20463/36051 [00:10<00:
07, 1976.19it/s]	57%		20670/36051 [00:10<00:
07, 1999.19it/s]	58%		20874/36051 [00:10<00:
07, 2006.92it/s]	58%		21076/36051 [00:11<00:
07, 1954.18it/s]	59%		21273/36051 [00:11<00:
07, 1914.84it/s]	60%		21468/36051 [00:11<00:
07, 1921.09it/s]	60%		21661/36051 [00:11<00:
07, 1874.94it/s]	61%		21850/36051 [00:11<00:
07, 1826.54it/s]	61%		22052/36051 [00:11<00:
07, 1876.60it/s]	62%		22253/36051 [00:11<00:
07, 1910.89it/s]	62%		22445/36051 [00:11<00:
07, 1777.23it/s]	63%		22626/36051 [00:12<00:
09, 1470.92it/s]	63%		22784/36051 [00:12<00:
09, 1352.97it/s]	64%		22929/36051 [00:12<00:
10, 1266.21it/s]			

64%		23064/36051	[00:12<00:
10,	1224.64it/s]		
64%		23193/36051	[00:12<00:
11,	1139.28it/s]		
65%		23319/36051	[00:12<00:
10,	1170.56it/s]		
65%		23490/36051	[00:12<00:
09,	1290.82it/s]		
66%		23647/36051	[00:12<00:
09,	1360.97it/s]		
66%		23789/36051	[00:12<00:
09,	1321.61it/s]		
66%		23966/36051	[00:13<00:
08,	1427.79it/s]		
67%		24175/36051	[00:13<00:
07,	1575.15it/s]		
68%		24387/36051	[00:13<00:
06,	1703.73it/s]		
68%		24606/36051	[00:13<00:
06,	1822.00it/s]		
69%		24825/36051	[00:13<00:
05,	1915.05it/s]		
69%		25028/36051	[00:13<00:
05,	1944.05it/s]		
70%		25228/36051	[00:13<00:
05,	1939.27it/s]		
71%		25460/36051	[00:13<00:
05,	2035.78it/s]		
71%		25704/36051	[00:13<00:
04,	2138.13it/s]		
72%		25922/36051	[00:14<00:
05,	1819.08it/s]		
72%		26115/36051	[00:14<00:
05,	1728.33it/s]		
73%		26297/36051	[00:14<00:
06,	1603.43it/s]		
73%		26466/36051	[00:14<00:
05,	1606.53it/s]		
74%		26633/36051	[00:14<00:
05,	1621.61it/s]		
74%		26811/36051	[00:14<00:
05,	1662.68it/s]		
75%		27040/36051	[00:14<00:
04,	1808.30it/s]		
76%		27276/36051	[00:14<00:
04,	1941.31it/s]		
76%		27482/36051	[00:14<00:
04,	1971.32it/s]		
77%		27685/36051	[00:14<00:
04,	1927.92it/s]		
77%		27884/36051	[00:15<00:
04,	1941.97it/s]		
78%		28093/36051	[00:15<00:
04,	1980.00it/s]		
78%		28293/36051	[00:15<00:
04,	1924.60it/s]		
79%		28488/36051	[00:15<00:
03,	1927.97it/s]		
80%		28720/36051	[00:15<00:
03,	2027.02it/s]		
80%		28945/36051	[00:15<00:
03,	2084.89it/s]		
81%		29175/36051	[00:15<00:
03,	2140.71it/s]		
82%		29391/36051	[00:15<00:
03,	2114.51it/s]		
82%		29609/36051	[00:15<00:
03,	2129.17it/s]		
83%		29823/36051	[00:16<00:
02,	2078.32it/s]		
83%		30044/36051	[00:16<00:
02,	2111.71it/s]		
84%		30275/36051	[00:16<00:
02,	2163.07it/s]		
85%		30493/36051	[00:16<00:
02,	2163.40it/s]		
85%		30718/36051	[00:16<00:

02, 2183.84it/s]		
86%		30944/36051 [00:16<00:
02, 2201.62it/s]		
86%		31172/36051 [00:16<00:
02, 2219.81it/s]		
87%		31407/36051 [00:16<00:
02, 2252.60it/s]		
88%		31634/36051 [00:16<00:
01, 2255.33it/s]		
88%		31863/36051 [00:16<00:
01, 2260.73it/s]		
89%		32095/36051 [00:17<00:
01, 2273.29it/s]		
90%		32331/36051 [00:17<00:
01, 2293.73it/s]		
90%		32568/36051 [00:17<00:
01, 2310.94it/s]		
91%		32800/36051 [00:17<00:
01, 2301.95it/s]		
92%		33031/36051 [00:17<00:
01, 2285.52it/s]		
92%		33268/36051 [00:17<00:
01, 2305.32it/s]		
93%		33501/36051 [00:17<00:
01, 2307.65it/s]		
94%		33736/36051 [00:17<00:
00, 2315.18it/s]		
94%		33970/36051 [00:17<00:
00, 2317.71it/s]		
95%		34202/36051 [00:17<00:
00, 2292.82it/s]		
96%		34432/36051 [00:18<00:
00, 2289.78it/s]		
96%		34674/36051 [00:18<00:
00, 2322.61it/s]		
97%		34916/36051 [00:18<00:
00, 2346.03it/s]		
98%		35151/36051 [00:18<00:
00, 2300.74it/s]		
98%		35382/36051 [00:18<00:
00, 2271.62it/s]		
99%		35612/36051 [00:18<00:
00, 2275.10it/s]		
99%		35840/36051 [00:18<00:
00, 2271.62it/s]		
100%		36051/36051 [00:18<00:
00, 1923.32it/s]		

In [279]:

```
avgw2vec_train_title=avgw2vectitle(x_train['title'].values)
avgw2vec_cv_title=avgw2vectitle(x_cv['title'].values)
avgw2vec_test_title=avgw2vectitle(x_test['title'].values)
```

In [282]:

```
from scipy.sparse import hstack

x_tr_word=hstack((avgw2vec_train_title,avgw2vec_train_essays,x_train_clean_subcategories_ohe,x_train_clean_categories_ohe,x_train_project_grade_category_ohe,x_train_teacher_ohe,x_train_state_ohe,x_train_prev_subj_stand,x_train_quan_stand,x_train_price_stand)).tocsr()
x_te_word=hstack((avgw2vec_test_title,avgw2vec_test_essays,x_test_clean_subcategories_ohe,x_test_clean_categories_ohe,x_test_project_grade_category_ohe,x_test_teacher_ohe,x_test_state_ohe,x_test_prev_subj_stand,x_test_quan_stand,x_test_price_stand)).tocsr()
x_cv_word=hstack((avgw2vec_cv_title,avgw2vec_cv_essays,x_cv_clean_subcategories_ohe,x_cv_clean_categories_ohe,x_cv_project_grade_category_ohe,x_cv_teacher_ohe,x_cv_state_ohe,x_cv_prev_subj_stand,x_cv_quan_stand,x_cv_price_stand)).tocsr()
print("Final Data matrix")
print(x_tr_word.shape, y_train.shape)
print(x_cv_word.shape, y_cv.shape)
print(x_te_word.shape, y_test.shape)
print("=="*100)
```

```
Final Data matrix
(49039, 702) (49039,)
(24155, 702) (24155,)
(36051, 702) (36051,)
```

In [287]:

```
#Hyperparameter tuning on Averagew2vec Model for C parameter

from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc

train_auc_word=[]
cv_auc_word=[]

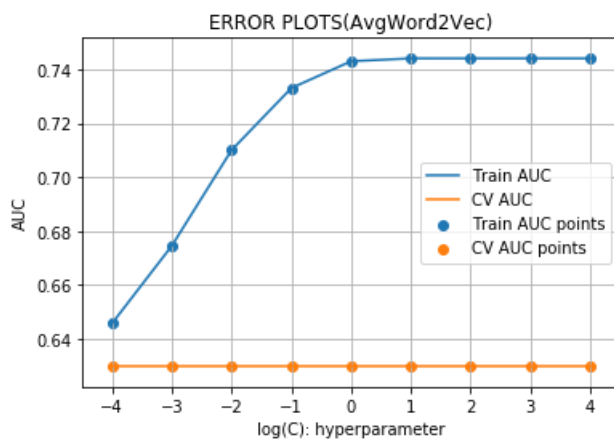
c_list=[10**(-4),10**(-3),10**(-2),10**(-1),10**(0),10**(1),10**(2),10**(3),10**(4)]
for i in c_list:
    clf=LogisticRegression(C=i,class_weight='balanced')
    clf.fit(x_tr_word,y_train)
    y_train_pred=clf.predict_proba(x_tr_word)[:,-1]
    y_cv_pred =clf.predict_proba(x_cv_word)[:,-1]
    train_auc_word.append(roc_auc_score(y_train,y_train_pred))
    cv_auc_word.append(roc_auc_score(y_cv,y_cv_pred))
```

In [305]:

```
plt.plot(np.log10(c_list), train_auc_word, label='Train AUC')
plt.plot(np.log10(c_list), cv_auc_word, label='CV AUC')

plt.scatter(np.log10(c_list), train_auc_word, label='Train AUC points')
plt.scatter(np.log10(c_list), cv_auc_word, label='CV AUC points')

plt.legend()
plt.xlabel("log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS (AvgWord2Vec)")
plt.grid()
plt.show()
```



In [301]:

```
best_c=10**0

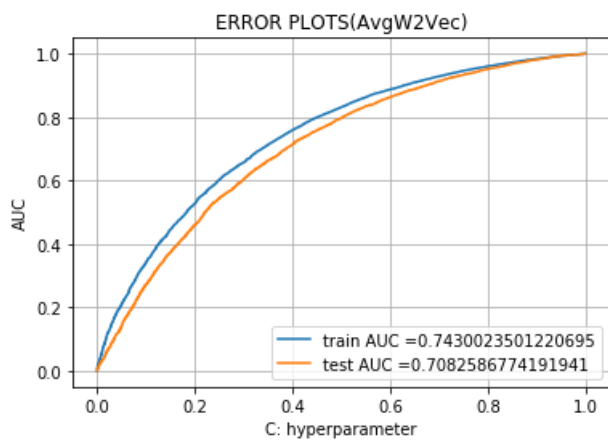
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression

clf_word=LogisticRegression(C=best_c,class_weight='balanced')
clf_word.fit(x_tr_word,y_train)

y_train_pred_word=clf_word.predict_proba(x_tr_word)[:,-1]
y_test_pred_word=clf_word.predict_proba(x_te_word)[:,-1]

train_fpr_word, train_tpr_word, tr_thresholds_word = roc_curve(y_train, y_train_pred_word)
test_fpr_word, test_tpr_word, te_thresholds_word = roc_curve(y_test, y_test_pred_word)
```

```
plt.plot(train_fpr_word, train_tpr_word, label="train AUC =" + str(auc(train_fpr_word, train_tpr_word)))
plt.plot(test_fpr_word, test_tpr_word, label="test AUC =" + str(auc(test_fpr_word, test_tpr_word)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS (AvgW2Vec)")
plt.grid()
plt.show()
```



In [548]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm_train=confusion_matrix(y_train, predict(y_train_pred_word, tr_thresholds_word, train_fpr_word, train_fpr_word))
print("Test confusion matrix")
cm_test=confusion_matrix(y_test, predict(y_test_pred_word, tr_thresholds_word, test_fpr_word, test_fpr_word))
```

Train confusion matrix
the maximum value of $tpr \cdot (1 - fpr)$ 0.24999999546531537 for threshold 0.396
Test confusion matrix
the maximum value of $tpr \cdot (1 - fpr)$ 0.24999999161092998 for threshold 0.47

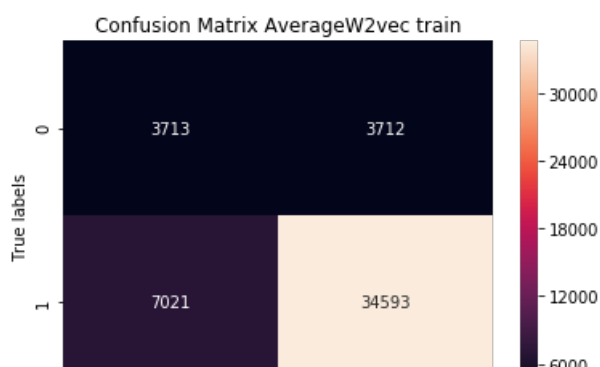
In [549]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(cm_train, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
print(y_train.value_counts())
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix AverageW2vec train');
```

```
1    41614
0     7425
Name: project_is_approved, dtype: int64
```





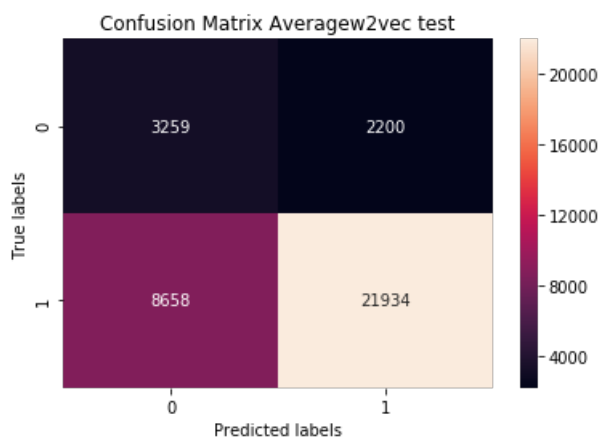
In [550]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(cm_test, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
print(y_test.value_counts())
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix Averagew2vec test');
```

```
1    30592
0     5459
Name: project_is_approved, dtype: int64
```



Logistic Regression with TFIDFWeightedWord2Vec representation of text

In [310]:

```
#tfidf weighted Word2vec for essays

def tfidfessays(preprocessed_essays):
    from tqdm import tqdm
    tfidf_w2v1_essays = []; # the avg-w2v for each sentence/review is stored in this list
    for sentence in tqdm(preprocessed_essays): # 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
        tfidf_w2v1_essays.append(vector)
    return tfidf_w2v1_essays
```

In [311]:

```
def tfidftitle(preprocessed_title):
    from tqdm import tqdm
    tfidf_w2v1_title = []; # the avg-w2v for each sentence/review is stored in this list
    for sentence in tqdm(preprocessed_title): # 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
    tfidf_w2v1_title.append(vector)
return tfidf_w2v1_title

```

In [312]:

```

tfidf_w2vec_essays_train=tfidfessays(x_train['essay'].values)
tfidf_w2vec_essays_cv=tfidfessays(x_cv['essay'].values)
tfidf_w2vec_essays_test=tfidfessays(x_test['essay'].values)

```

```

0%|          | 0/49039
[00:00<?, ?it/s]
0%|          | 26/49039 [00:00<03
:11, 256.42it/s]
0%||         | 93/49039 [00:00<02
:35, 314.64it/s]
0%||         | 189/49039 [00:00<02
:04, 392.24it/s]
1%||         | 292/49039 [00:00<01
:41, 480.35it/s]
1%||         | 389/49039 [00:00<01
:25, 565.71it/s]
1%||         | 462/49039 [00:00<01
:22, 587.15it/s]
1%||         | 548/49039 [00:00<01
:14, 648.61it/s]
1%||         | 666/49039 [00:00<01
:04, 747.91it/s]
2%||         | 787/49039 [00:00<00
:57, 838.35it/s]
2%||         | 916/49039 [00:01<00
:51, 936.11it/s]
2%||         | 1039/49039 [00:01<00:
47, 1007.86it/s]
2%||         | 1158/49039 [00:01<00:
45, 1050.78it/s]
3%||         | 1298/49039 [00:01<00:
42, 1127.33it/s]
3%||         | 1425/49039 [00:01<00:
40, 1166.43it/s]
3%||         | 1574/49039 [00:01<00:
38, 1241.11it/s]
3%||         | 1713/49039 [00:01<00:
36, 1280.33it/s]
4%||         | 1846/49039 [00:01<00:
36, 1293.71it/s]
4%||         | 1986/49039 [00:01<00:
35, 1314.10it/s]
4%||         | 2127/49039 [00:01<00:
35, 1337.37it/s]
5%||         | 2275/49039 [00:02<00:
34, 1368.10it/s]
5%||         | 2436/49039 [00:02<00:
32, 1423.99it/s]
5%||         | 2580/49039 [00:02<00:
33, 1403.99it/s]
6%||         | 2722/49039 [00:02<00:
33, 1389.59it/s]
6%||         | 2865/49039 [00:02<00:
32, 1400.41it/s]
6%||         | 3019/49039 [00:02<00:
32, 1423.81it/s]
6%||         | 3168/49039 [00:02<00:
31, 1440.91it/s]
7%||         | 3313/49039 [00:02<00:
31, 1438.69it/s]
7%||         | 3462/49039 [00:02<00:
31, 1445.13it/s]
7%||         | 3607/49039 [00:02<00:
31, 1434.91it/s]
8%||         | 3759/49039 [00:03<00:

```

31, 1445.77it/s]	8% ██████████	3933/49039 [00:03<00:
29, 1520.28it/s]	8% ██████████	4087/49039 [00:03<00:
30, 1488.49it/s]	9% ██████████	4237/49039 [00:03<00:
30, 1487.91it/s]	9% ██████████	4387/49039 [00:03<00:
29, 1488.63it/s]	9% ██████████	4546/49039 [00:03<00:
29, 1517.53it/s]	10% ██████████	4707/49039 [00:03<00:
28, 1529.78it/s]	10% ██████████	4880/49039 [00:03<00:
28, 1576.46it/s]	10% ██████████	5053/49039 [00:03<00:
27, 1605.12it/s]	11% ██████████	5215/49039 [00:03<00:
27, 1597.13it/s]	11% ██████████	5382/49039 [00:04<00:
27, 1604.32it/s]	11% ██████████	5551/49039 [00:04<00:
26, 1621.31it/s]	12% ██████████	5714/49039 [00:04<00:
27, 1601.81it/s]	12% ██████████	5887/49039 [00:04<00:
26, 1636.44it/s]	12% ██████████	6053/49039 [00:04<00:
26, 1641.74it/s]	13% ██████████	6221/49039 [00:04<00:
25, 1650.80it/s]	13% ██████████	6391/49039 [00:04<00:
25, 1662.91it/s]	13% ██████████	6577/49039 [00:04<00:
24, 1712.92it/s]	14% ██████████	6749/49039 [00:04<00:
24, 1705.04it/s]	14% ██████████	6922/49039 [00:05<00:
24, 1707.42it/s]	14% ██████████	7093/49039 [00:05<00:
24, 1697.43it/s]	15% ██████████	7263/49039 [00:05<00:
28, 1440.92it/s]	15% ██████████	7414/49039 [00:05<00:
28, 1460.04it/s]	15% ██████████	7576/49039 [00:05<00:
27, 1494.49it/s]	16% ██████████	7759/49039 [00:05<00:
26, 1572.38it/s]	16% ██████████	7924/49039 [00:05<00:
25, 1585.12it/s]	17% ██████████	8104/49039 [00:05<00:
25, 1632.07it/s]	17% ██████████	8274/49039 [00:05<00:
24, 1645.88it/s]	17% ██████████	8441/49039 [00:05<00:
24, 1636.52it/s]	18% ██████████	8622/49039 [00:06<00:
24, 1676.17it/s]	18% ██████████	8793/49039 [00:06<00:
23, 1677.97it/s]	18% ██████████	8968/49039 [00:06<00:
23, 1687.09it/s]	19% ██████████	9138/49039 [00:06<00:
23, 1675.66it/s]	19% ██████████	9320/49039 [00:06<00:
23, 1699.60it/s]	19% ██████████	9497/49039 [00:06<00:
23, 1711.32it/s]	20% ██████████	9669/49039 [00:06<00:
23, 1650.22it/s]	20% ██████████	9847/49039 [00:06<00:
23, 1671.85it/s]	20% ██████████	10039/49039 [00:06<00:
22, 1720.68it/s]	21% ██████████	10227/49039 [00:07<00:
22, 1760.85it/s]	██████████	

21% [REDACTED]	10412/49039 [00:07<00:
21, 1770.26it/s]	
22% [REDACTED]	10590/49039 [00:07<00:
22, 1722.19it/s]	
22% [REDACTED]	10763/49039 [00:07<00:
22, 1701.44it/s]	
22% [REDACTED]	10934/49039 [00:07<00:
22, 1694.07it/s]	
23% [REDACTED]	11116/49039 [00:07<00:
22, 1722.92it/s]	
23% [REDACTED]	11290/49039 [00:07<00:
22, 1707.76it/s]	
23% [REDACTED]	11462/49039 [00:07<00:
21, 1711.00it/s]	
24% [REDACTED]	11634/49039 [00:07<00:
22, 1670.04it/s]	
24% [REDACTED]	11802/49039 [00:07<00:
24, 1505.66it/s]	
24% [REDACTED]	11956/49039 [00:08<00:
24, 1507.50it/s]	
25% [REDACTED]	12110/49039 [00:08<00:
25, 1446.83it/s]	
25% [REDACTED]	12257/49039 [00:08<00:
25, 1423.41it/s]	
25% [REDACTED]	12420/49039 [00:08<00:
24, 1479.40it/s]	
26% [REDACTED]	12591/49039 [00:08<00:
23, 1526.48it/s]	
26% [REDACTED]	12760/49039 [00:08<00:
23, 1571.32it/s]	
26% [REDACTED]	12953/49039 [00:08<00:
21, 1647.48it/s]	
27% [REDACTED]	13120/49039 [00:08<00:
21, 1644.48it/s]	
27% [REDACTED]	13291/49039 [00:08<00:
21, 1657.67it/s]	
27% [REDACTED]	13460/49039 [00:09<00:
21, 1661.10it/s]	
28% [REDACTED]	13639/49039 [00:09<00:
20, 1691.80it/s]	
28% [REDACTED]	13820/49039 [00:09<00:
20, 1716.10it/s]	
29% [REDACTED]	13999/49039 [00:09<00:
20, 1736.90it/s]	
29% [REDACTED]	14188/49039 [00:09<00:
19, 1763.58it/s]	
29% [REDACTED]	14369/49039 [00:09<00:
19, 1762.57it/s]	
30% [REDACTED]	14553/49039 [00:09<00:
19, 1782.74it/s]	
30% [REDACTED]	14746/49039 [00:09<00:
18, 1812.75it/s]	
30% [REDACTED]	14928/49039 [00:09<00:
19, 1748.18it/s]	
31% [REDACTED]	15104/49039 [00:09<00:
21, 1613.70it/s]	
31% [REDACTED]	15268/49039 [00:10<00:
21, 1604.58it/s]	
31% [REDACTED]	15436/49039 [00:10<00:
20, 1620.96it/s]	
32% [REDACTED]	15631/49039 [00:10<00:
19, 1689.94it/s]	
32% [REDACTED]	15810/49039 [00:10<00:
19, 1703.89it/s]	
33% [REDACTED]	15989/49039 [00:10<00:
19, 1717.11it/s]	
33% [REDACTED]	16162/49039 [00:10<00:
19, 1688.00it/s]	
33% [REDACTED]	16332/49039 [00:10<00:
19, 1671.42it/s]	
34% [REDACTED]	16510/49039 [00:10<00:
19, 1684.21it/s]	
34% [REDACTED]	16689/49039 [00:10<00:
18, 1703.78it/s]	
34% [REDACTED]	16860/49039 [00:11<00:
19, 1666.99it/s]	
35% [REDACTED]	17046/49039 [00:11<00:

18, 1709.78it/s]	35%	17228/49039 [00:11<00:
18, 1734.76it/s]	35%	17402/49039 [00:11<00:
18, 1711.62it/s]	36%	17574/49039 [00:11<00:
18, 1709.04it/s]	36%	17746/49039 [00:11<00:
18, 1693.40it/s]	37%	17916/49039 [00:11<00:
18, 1693.55it/s]	37%	18103/49039 [00:11<00:
17, 1732.76it/s]	37%	18280/49039 [00:11<00:
17, 1743.59it/s]	38%	18472/49039 [00:11<00:
17, 1790.41it/s]	38%	18652/49039 [00:12<00:
17, 1765.48it/s]	38%	18829/49039 [00:12<00:
17, 1742.71it/s]	39%	19004/49039 [00:12<00:
18, 1605.37it/s]	39%	19167/49039 [00:12<00:
19, 1565.88it/s]	39%	19362/49039 [00:12<00:
18, 1643.26it/s]	40%	19529/49039 [00:12<00:
20, 1471.92it/s]	40%	19709/49039 [00:12<00:
18, 1547.95it/s]	41%	19882/49039 [00:12<00:
18, 1590.92it/s]	41%	20063/49039 [00:12<00:
17, 1646.45it/s]	41%	20231/49039 [00:13<00:
17, 1621.86it/s]	42%	20396/49039 [00:13<00:
18, 1534.38it/s]	42%	20590/49039 [00:13<00:
17, 1622.15it/s]	42%	20765/49039 [00:13<00:
17, 1643.34it/s]	43%	20950/49039 [00:13<00:
16, 1685.72it/s]	43%	21127/49039 [00:13<00:
16, 1709.36it/s]	43%	21300/49039 [00:13<00:
18, 1515.32it/s]	44%	21480/49039 [00:13<00:
17, 1584.71it/s]	44%	21643/49039 [00:13<00:
18, 1476.06it/s]	45%	21832/49039 [00:14<00:
17, 1578.88it/s]	45%	22019/49039 [00:14<00:
16, 1640.16it/s]	45%	22188/49039 [00:14<00:
16, 1622.22it/s]	46%	22373/49039 [00:14<00:
15, 1682.25it/s]	46%	22564/49039 [00:14<00:
15, 1727.34it/s]	46%	22739/49039 [00:14<00:
15, 1710.24it/s]	47%	22912/49039 [00:14<00:
15, 1711.76it/s]	47%	23104/49039 [00:14<00:
14, 1761.97it/s]	47%	23282/49039 [00:14<00:
17, 1500.63it/s]	48%	23445/49039 [00:15<00:
16, 1531.16it/s]	48%	23620/49039 [00:15<00:
15, 1590.13it/s]	49%	23784/49039 [00:15<00:
15, 1601.29it/s]		

49%		23948/49039	[00:15<00:
18, 1392.22it/s]			
49%		24115/49039	[00:15<00:
17, 1464.11it/s]			
50%		24297/49039	[00:15<00:
15, 1553.77it/s]			
50%		24488/49039	[00:15<00:
15, 1634.82it/s]			
50%		24661/49039	[00:15<00:
14, 1661.50it/s]			
51%		24835/49039	[00:15<00:
14, 1683.94it/s]			
51%		25007/49039	[00:16<00:
15, 1531.74it/s]			
51%		25174/49039	[00:16<00:
15, 1554.72it/s]			
52%		25333/49039	[00:16<00:
15, 1562.81it/s]			
52%		25516/49039	[00:16<00:
14, 1628.74it/s]			
52%		25687/49039	[00:16<00:
14, 1651.16it/s]			
53%		25869/49039	[00:16<00:
13, 1679.84it/s]			
53%		26046/49039	[00:16<00:
13, 1701.29it/s]			
53%		26227/49039	[00:16<00:
13, 1718.30it/s]			
54%		26409/49039	[00:16<00:
12, 1742.42it/s]			
54%		26584/49039	[00:16<00:
13, 1698.96it/s]			
55%		26755/49039	[00:17<00:
16, 1374.18it/s]			
55%		26927/49039	[00:17<00:
15, 1457.20it/s]			
55%		27114/49039	[00:17<00:
14, 1550.61it/s]			
56%		27293/49039	[00:17<00:
13, 1614.10it/s]			
56%		27461/49039	[00:17<00:
13, 1627.91it/s]			
56%		27629/49039	[00:17<00:
13, 1606.54it/s]			
57%		27807/49039	[00:17<00:
12, 1641.73it/s]			
57%		27987/49039	[00:17<00:
12, 1675.69it/s]			
57%		28162/49039	[00:17<00:
12, 1682.10it/s]			
58%		28332/49039	[00:18<00:
13, 1529.76it/s]			
58%		28521/49039	[00:18<00:
12, 1622.40it/s]			
59%		28712/49039	[00:18<00:
11, 1694.23it/s]			
59%		28886/49039	[00:18<00:
12, 1672.35it/s]			
59%		29056/49039	[00:18<00:
11, 1677.84it/s]			
60%		29237/49039	[00:18<00:
11, 1694.83it/s]			
60%		29408/49039	[00:18<00:
11, 1691.19it/s]			
60%		29602/49039	[00:18<00:
11, 1747.81it/s]			
61%		29791/49039	[00:18<00:
10, 1786.85it/s]			
61%		29979/49039	[00:19<00:
10, 1794.17it/s]			
62%		30163/49039	[00:19<00:
10, 1802.36it/s]			
62%		30344/49039	[00:19<00:
10, 1717.44it/s]			
62%		30517/49039	[00:19<00:
11, 1659.23it/s]			
63%		30707/49039	[00:19<00:

10, 1712.24it/s]	63%		30883/49039 [00:19<00:
10, 1720.70it/s]	63%		31071/49039 [00:19<00:
10, 1754.10it/s]	64%		31272/49039 [00:19<00:
09, 1813.52it/s]	64%		31455/49039 [00:19<00:
09, 1794.55it/s]	65%		31636/49039 [00:19<00:
09, 1747.88it/s]	65%		31812/49039 [00:20<00:
09, 1726.44it/s]	65%		31986/49039 [00:20<00:
09, 1705.53it/s]	66%		32164/49039 [00:20<00:
09, 1725.57it/s]	66%		32338/49039 [00:20<00:
09, 1724.31it/s]	66%		32533/49039 [00:20<00:
09, 1770.45it/s]	67%		32711/49039 [00:20<00:
09, 1747.79it/s]	67%		32889/49039 [00:20<00:
09, 1751.84it/s]	67%		33075/49039 [00:20<00:
08, 1778.03it/s]	68%		33266/49039 [00:20<00:
08, 1796.09it/s]	68%		33461/49039 [00:20<00:
08, 1835.58it/s]	69%		33646/49039 [00:21<00:
08, 1833.53it/s]	69%		33831/49039 [00:21<00:
08, 1828.99it/s]	69%		34021/49039 [00:21<00:
08, 1843.89it/s]	70%		34206/49039 [00:21<00:
08, 1818.26it/s]	70%		34389/49039 [00:21<00:
08, 1808.31it/s]	71%		34577/49039 [00:21<00:
07, 1815.78it/s]	71%		34759/49039 [00:21<00:
08, 1731.99it/s]	71%		34934/49039 [00:21<00:
08, 1710.66it/s]	72%		35119/49039 [00:21<00:
07, 1740.44it/s]	72%		35294/49039 [00:22<00:
08, 1623.62it/s]	72%		35463/49039 [00:22<00:
08, 1642.00it/s]	73%		35630/49039 [00:22<00:
08, 1639.93it/s]	73%		35801/49039 [00:22<00:
08, 1651.86it/s]	73%		35967/49039 [00:22<00:
07, 1641.35it/s]	74%		36143/49039 [00:22<00:
07, 1660.27it/s]	74%		36329/49039 [00:22<00:
07, 1699.05it/s]	74%		36510/49039 [00:22<00:
07, 1727.23it/s]	75%		36685/49039 [00:22<00:
07, 1717.19it/s]	75%		36858/49039 [00:22<00:
07, 1594.70it/s]	76%		37030/49039 [00:23<00:
07, 1611.08it/s]	76%		37193/49039 [00:23<00:
07, 1607.58it/s]	76%		37382/49039 [00:23<00:
06, 1668.23it/s]	77%		37566/49039 [00:23<00:
06, 1704.00it/s]			

77%		37738/49039 [00:23<00:
06, 1704.37it/s]		
77%		37919/49039 [00:23<00:
06, 1719.38it/s]		
78%		38110/49039 [00:23<00:
06, 1758.29it/s]		
78%		38292/49039 [00:23<00:
06, 1770.33it/s]		
78%		38470/49039 [00:23<00:
05, 1762.82it/s]		
79%		38647/49039 [00:24<00:
06, 1607.40it/s]		
79%		38823/49039 [00:24<00:
06, 1646.31it/s]		
80%		38990/49039 [00:24<00:
06, 1603.46it/s]		
80%		39169/49039 [00:24<00:
05, 1654.17it/s]		
80%		39337/49039 [00:24<00:
06, 1492.79it/s]		
81%		39499/49039 [00:24<00:
06, 1528.73it/s]		
81%		39673/49039 [00:24<00:
05, 1577.41it/s]		
81%		39857/49039 [00:24<00:
05, 1646.83it/s]		
82%		40048/49039 [00:24<00:
05, 1699.87it/s]		
82%		40228/49039 [00:24<00:
05, 1725.91it/s]		
82%		40403/49039 [00:25<00:
05, 1712.05it/s]		
83%		40576/49039 [00:25<00:
05, 1553.27it/s]		
83%		40743/49039 [00:25<00:
05, 1583.67it/s]		
83%		40921/49039 [00:25<00:
04, 1631.35it/s]		
84%		41104/49039 [00:25<00:
04, 1682.93it/s]		
84%		41275/49039 [00:25<00:
04, 1633.72it/s]		
85%		41453/49039 [00:25<00:
04, 1662.80it/s]		
85%		41630/49039 [00:25<00:
04, 1690.85it/s]		
85%		41812/49039 [00:25<00:
04, 1719.83it/s]		
86%		41985/49039 [00:26<00:
04, 1669.65it/s]		
86%		42153/49039 [00:26<00:
04, 1578.54it/s]		
86%		42320/49039 [00:26<00:
04, 1598.49it/s]		
87%		42508/49039 [00:26<00:
03, 1658.26it/s]		
87%		42698/49039 [00:26<00:
03, 1718.78it/s]		
87%		42872/49039 [00:26<00:
04, 1533.95it/s]		
88%		43053/49039 [00:26<00:
03, 1598.78it/s]		
88%		43242/49039 [00:26<00:
03, 1671.51it/s]		
89%		43422/49039 [00:26<00:
03, 1697.88it/s]		
89%		43597/49039 [00:27<00:
03, 1708.94it/s]		
89%		43784/49039 [00:27<00:
02, 1753.67it/s]		
90%		43961/49039 [00:27<00:
02, 1704.81it/s]		
90%		44146/49039 [00:27<00:
02, 1744.79it/s]		
90%		44322/49039 [00:27<00:
02, 1659.88it/s]		
91%		44490/49039 [00:27<00:

02, 1531.76it/s]		
91%		44653/49039 [00:27<00:
02, 1556.47it/s]		
91%		44853/49039 [00:27<00:
02, 1663.90it/s]		
92%		45024/49039 [00:27<00:
02, 1666.62it/s]		
92%		45198/49039 [00:28<00:
02, 1682.52it/s]		
93%		45384/49039 [00:28<00:
02, 1730.73it/s]		
93%		45559/49039 [00:28<00:
02, 1713.19it/s]		
93%		45741/49039 [00:28<00:
01, 1729.14it/s]		
94%		45934/49039 [00:28<00:
01, 1765.72it/s]		
94%		46131/49039 [00:28<00:
01, 1803.80it/s]		
94%		46313/49039 [00:28<00:
01, 1807.58it/s]		
95%		46495/49039 [00:28<00:
01, 1802.88it/s]		
95%		46685/49039 [00:28<00:
01, 1827.51it/s]		
96%		46869/49039 [00:28<00:
01, 1828.66it/s]		
96%		47053/49039 [00:29<00:
01, 1805.61it/s]		
96%		47234/49039 [00:29<00:
01, 1675.50it/s]		
97%		47404/49039 [00:29<00:
00, 1656.13it/s]		
97%		47594/49039 [00:29<00:
00, 1718.92it/s]		
97%		47772/49039 [00:29<00:
00, 1726.11it/s]		
98%		47946/49039 [00:29<00:
00, 1472.51it/s]		
98%		48101/49039 [00:29<00:
00, 1289.10it/s]		
98%		48256/49039 [00:29<00:
00, 1354.70it/s]		
99%		48440/49039 [00:29<00:
00, 1468.40it/s]		
99%		48627/49039 [00:30<00:
00, 1557.63it/s]		
99%		48790/49039 [00:30<00:
00, 1505.79it/s]		
100%		48981/49039 [00:30<00:
00, 1606.02it/s]		
100%		49039/49039 [00:30<00:
00, 1616.61it/s]		
0%		0/24155
[00:00<?, ?it/s]		
1%		174/24155 [00:00<00:
13, 1730.17it/s]		
1%		354/24155 [00:00<00:
13, 1737.46it/s]		
2%		536/24155 [00:00<00:
13, 1760.83it/s]		
3%		653/24155 [00:00<00:
15, 1483.62it/s]		
3%		807/24155 [00:00<00:
15, 1488.40it/s]		
4%		999/24155 [00:00<00:
14, 1593.06it/s]		
5%		1192/24155 [00:00<00:
13, 1668.14it/s]		
6%		1374/24155 [00:00<00:
13, 1700.25it/s]		
6%		1538/24155 [00:00<00:
13, 1615.66it/s]		
7%		1718/24155 [00:01<00:
13, 1653.60it/s]		
8%		1917/24155 [00:01<00:
12, 1726.37it/s]		

9% [REDACTED]	2089/24155 [00:01<00:
13, 1630.22it/s]	
9% [REDACTED]	2263/24155 [00:01<00:
13, 1646.37it/s]	
10% [REDACTED]	2429/24155 [00:01<00:
13, 1623.24it/s]	
11% [REDACTED]	2600/24155 [00:01<00:
13, 1640.26it/s]	
11% [REDACTED]	2765/24155 [00:01<00:
14, 1507.17it/s]	
12% [REDACTED]	2927/24155 [00:01<00:
13, 1528.29it/s]	
13% [REDACTED]	3112/24155 [00:01<00:
13, 1600.68it/s]	
14% [REDACTED]	3299/24155 [00:02<00:
12, 1670.78it/s]	
14% [REDACTED]	3469/24155 [00:02<00:
12, 1672.42it/s]	
15% [REDACTED]	3660/24155 [00:02<00:
11, 1736.25it/s]	
16% [REDACTED]	3843/24155 [00:02<00:
11, 1759.14it/s]	
17% [REDACTED]	4021/24155 [00:02<00:
13, 1535.09it/s]	
17% [REDACTED]	4181/24155 [00:02<00:
13, 1470.84it/s]	
18% [REDACTED]	4349/24155 [00:02<00:
13, 1517.64it/s]	
19% [REDACTED]	4531/24155 [00:02<00:
12, 1583.32it/s]	
19% [REDACTED]	4693/24155 [00:02<00:
12, 1588.31it/s]	
20% [REDACTED]	4855/24155 [00:03<00:
14, 1289.29it/s]	
21% [REDACTED]	5021/24155 [00:03<00:
13, 1371.43it/s]	
22% [REDACTED]	5201/24155 [00:03<00:
12, 1471.44it/s]	
22% [REDACTED]	5391/24155 [00:03<00:
11, 1571.89it/s]	
23% [REDACTED]	5556/24155 [00:03<00:
11, 1591.66it/s]	
24% [REDACTED]	5721/24155 [00:03<00:
13, 1322.98it/s]	
24% [REDACTED]	5865/24155 [00:03<00:
14, 1305.25it/s]	
25% [REDACTED]	6043/24155 [00:03<00:
12, 1405.45it/s]	
26% [REDACTED]	6219/24155 [00:03<00:
12, 1490.92it/s]	
27% [REDACTED]	6414/24155 [00:04<00:
11, 1588.89it/s]	
27% [REDACTED]	6594/24155 [00:04<00:
10, 1635.31it/s]	
28% [REDACTED]	6763/24155 [00:04<00:
10, 1643.71it/s]	
29% [REDACTED]	6949/24155 [00:04<00:
10, 1701.39it/s]	
29% [REDACTED]	7123/24155 [00:04<00:
10, 1702.69it/s]	
30% [REDACTED]	7296/24155 [00:04<00:
11, 1470.51it/s]	
31% [REDACTED]	7451/24155 [00:04<00:
11, 1461.98it/s]	
32% [REDACTED]	7652/24155 [00:04<00:
10, 1582.67it/s]	
32% [REDACTED]	7826/24155 [00:04<00:
10, 1623.48it/s]	
33% [REDACTED]	8008/24155 [00:05<00:
09, 1672.16it/s]	
34% [REDACTED]	8186/24155 [00:05<00:
09, 1701.97it/s]	
35% [REDACTED]	8359/24155 [00:05<00:
09, 1706.47it/s]	
35% [REDACTED]	8555/24155 [00:05<00:
08, 1763.21it/s]	
36% [REDACTED]	8734/24155 [00:05<00:

08, 1720.15it/s]		
37%		8910/24155 [00:05<00:
08, 1730.47it/s]		
38%		9094/24155 [00:05<00:
08, 1746.83it/s]		
38%		9282/24155 [00:05<00:
08, 1767.92it/s]		
39%		9460/24155 [00:05<00:
08, 1679.15it/s]		
40%		9660/24155 [00:05<00:
08, 1754.42it/s]		
41%		9838/24155 [00:06<00:
08, 1714.79it/s]		
41%		10011/24155 [00:06<00:
08, 1695.06it/s]		
42%		10190/24155 [00:06<00:
08, 1719.46it/s]		
43%		10363/24155 [00:06<00:
09, 1517.37it/s]		
44%		10547/24155 [00:06<00:
08, 1592.90it/s]		
44%		10740/24155 [00:06<00:
08, 1664.64it/s]		
45%		10918/24155 [00:06<00:
07, 1685.68it/s]		
46%		11090/24155 [00:06<00:
08, 1589.08it/s]		
47%		11253/24155 [00:06<00:
08, 1599.42it/s]		
47%		11416/24155 [00:07<00:
08, 1464.14it/s]		
48%		11589/24155 [00:07<00:
08, 1516.31it/s]		
49%		11768/24155 [00:07<00:
07, 1585.28it/s]		
49%		11930/24155 [00:07<00:
08, 1457.51it/s]		
50%		12097/24155 [00:07<00:
07, 1513.08it/s]		
51%		12261/24155 [00:07<00:
07, 1546.95it/s]		
51%		12419/24155 [00:07<00:
08, 1443.15it/s]		
52%		12589/24155 [00:07<00:
07, 1497.85it/s]		
53%		12742/24155 [00:08<00:
08, 1318.87it/s]		
53%		12880/24155 [00:08<00:
08, 1334.28it/s]		
54%		13049/24155 [00:08<00:
07, 1413.15it/s]		
55%		13195/24155 [00:08<00:
08, 1345.00it/s]		
55%		13343/24155 [00:08<00:
07, 1368.43it/s]		
56%		13546/24155 [00:08<00:
07, 1508.53it/s]		
57%		13719/24155 [00:08<00:
06, 1565.43it/s]		
58%		13893/24155 [00:08<00:
06, 1598.18it/s]		
58%		14057/24155 [00:08<00:
06, 1599.60it/s]		
59%		14220/24155 [00:08<00:
06, 1494.23it/s]		
60%		14373/24155 [00:09<00:
06, 1473.98it/s]		
60%		14531/24155 [00:09<00:
06, 1500.85it/s]		
61%		14683/24155 [00:09<00:
06, 1502.05it/s]		
62%		14862/24155 [00:09<00:
05, 1577.57it/s]		
62%		15053/24155 [00:09<00:
05, 1655.00it/s]		
63%		15247/24155 [00:09<00:
05, 1728.91it/s]		

64%		15423/24155 [00:09<00:
05, 1610.25it/s]		
65%		15588/24155 [00:09<00:
05, 1584.91it/s]		
65%		15749/24155 [00:09<00:
05, 1562.56it/s]		
66%		15919/24155 [00:10<00:
05, 1596.19it/s]		
67%		16091/24155 [00:10<00:
04, 1621.05it/s]		
67%		16256/24155 [00:10<00:
04, 1629.52it/s]		
68%		16433/24155 [00:10<00:
04, 1664.87it/s]		
69%		16601/24155 [00:10<00:
04, 1609.05it/s]		
69%		16763/24155 [00:10<00:
05, 1459.46it/s]		
70%		16921/24155 [00:10<00:
04, 1478.41it/s]		
71%		17100/24155 [00:10<00:
04, 1558.78it/s]		
72%		17284/24155 [00:10<00:
04, 1620.81it/s]		
72%		17449/24155 [00:11<00:
04, 1601.72it/s]		
73%		17617/24155 [00:11<00:
04, 1609.28it/s]		
74%		17780/24155 [00:11<00:
04, 1419.98it/s]		
74%		17928/24155 [00:11<00:
04, 1425.06it/s]		
75%		18107/24155 [00:11<00:
03, 1512.96it/s]		
76%		18263/24155 [00:11<00:
04, 1430.02it/s]		
76%		18410/24155 [00:11<00:
04, 1435.21it/s]		
77%		18602/24155 [00:11<00:
03, 1549.33it/s]		
78%		18764/24155 [00:11<00:
03, 1561.68it/s]		
78%		18934/24155 [00:11<00:
03, 1597.56it/s]		
79%		19106/24155 [00:12<00:
03, 1613.95it/s]		
80%		19295/24155 [00:12<00:
02, 1675.20it/s]		
81%		19465/24155 [00:12<00:
02, 1635.16it/s]		
81%		19634/24155 [00:12<00:
02, 1645.52it/s]		
82%		19800/24155 [00:12<00:
02, 1642.29it/s]		
83%		19982/24155 [00:12<00:
02, 1684.47it/s]		
83%		20152/24155 [00:12<00:
02, 1681.05it/s]		
84%		20340/24155 [00:12<00:
02, 1731.02it/s]		
85%		20523/24155 [00:12<00:
02, 1753.96it/s]		
86%		20699/24155 [00:13<00:
02, 1684.47it/s]		
86%		20869/24155 [00:13<00:
02, 1609.78it/s]		
87%		21062/24155 [00:13<00:
01, 1693.47it/s]		
88%		21234/24155 [00:13<00:
01, 1676.23it/s]		
89%		21404/24155 [00:13<00:
01, 1629.09it/s]		
89%		21579/24155 [00:13<00:
01, 1659.14it/s]		
90%		21747/24155 [00:13<00:
01, 1632.28it/s]		
91%		21912/24155 [00:13<00:

01, 1616.58it/s]		
91%	22075/24155	[00:13<00:
01, 1573.81it/s]		
92%	22239/24155	[00:13<00:
01, 1590.97it/s]		
93%	22415/24155	[00:14<00:
01, 1619.59it/s]		
93%	22578/24155	[00:14<00:
01, 1566.71it/s]		
94%	22736/24155	[00:14<00:
00, 1505.97it/s]		
95%	22888/24155	[00:14<00:
00, 1461.34it/s]		
95%	23059/24155	[00:14<00:
00, 1520.64it/s]		
96%	23213/24155	[00:14<00:
00, 1484.89it/s]		
97%	23407/24155	[00:14<00:
00, 1590.06it/s]		
98%	23595/24155	[00:14<00:
00, 1657.73it/s]		
98%	23766/24155	[00:14<00:
00, 1664.05it/s]		
99%	23940/24155	[00:15<00:
00, 1675.88it/s]		
100%	24117/24155	[00:15<00:
00, 1688.81it/s]		
100%	24155/24155	[00:15<00:
00, 1591.68it/s]		
0%		0/36051
[00:00<?, ?it/s]		
1%	189/36051	[00:00<00:
19, 1856.98it/s]		
1%	336/36051	[00:00<00:
20, 1720.94it/s]		
1%	483/36051	[00:00<00:
21, 1632.38it/s]		
2%	649/36051	[00:00<00:
21, 1639.04it/s]		
2%	823/36051	[00:00<00:
21, 1659.16it/s]		
3%	1005/36051	[00:00<00:
20, 1694.71it/s]		
3%	1182/36051	[00:00<00:
20, 1714.75it/s]		
4%	1371/36051	[00:00<00:
19, 1761.24it/s]		
4%	1553/36051	[00:00<00:
19, 1763.22it/s]		
5%	1723/36051	[00:01<00:
19, 1733.62it/s]		
5%	1892/36051	[00:01<00:
20, 1690.21it/s]		
6%	2058/36051	[00:01<00:
20, 1663.20it/s]		
6%	2223/36051	[00:01<00:
20, 1621.08it/s]		
7%	2392/36051	[00:01<00:
20, 1637.13it/s]		
7%	2575/36051	[00:01<00:
19, 1684.42it/s]		
8%	2744/36051	[00:01<00:
19, 1683.80it/s]		
8%	2920/36051	[00:01<00:
19, 1693.24it/s]		
9%	3090/36051	[00:01<00:
20, 1645.52it/s]		
9%	3255/36051	[00:01<00:
20, 1625.54it/s]		
9%	3418/36051	[00:02<00:
20, 1614.62it/s]		
10%	3580/36051	[00:02<00:
20, 1602.64it/s]		
10%	3751/36051	[00:02<00:
19, 1633.08it/s]		
11%	3934/36051	[00:02<00:
19, 1673.80it/s]		

11% [REDACTED]	4106/36051 [00:02<00:
18, 1683.58it/s]	
12% [REDACTED]	4275/36051 [00:02<00:
18, 1685.39it/s]	
12% [REDACTED]	4444/36051 [00:02<00:
18, 1679.85it/s]	
13% [REDACTED]	4625/36051 [00:02<00:
18, 1713.39it/s]	
13% [REDACTED]	4803/36051 [00:02<00:
18, 1714.37it/s]	
14% [REDACTED]	4990/36051 [00:02<00:
17, 1752.89it/s]	
14% [REDACTED]	5166/36051 [00:03<00:
18, 1701.00it/s]	
15% [REDACTED]	5337/36051 [00:03<00:
18, 1635.58it/s]	
15% [REDACTED]	5502/36051 [00:03<00:
19, 1601.54it/s]	
16% [REDACTED]	5663/36051 [00:03<00:
19, 1555.40it/s]	
16% [REDACTED]	5820/36051 [00:03<00:
20, 1505.39it/s]	
17% [REDACTED]	5986/36051 [00:03<00:
19, 1538.47it/s]	
17% [REDACTED]	6142/36051 [00:03<00:
19, 1540.90it/s]	
17% [REDACTED]	6302/36051 [00:03<00:
19, 1556.90it/s]	
18% [REDACTED]	6464/36051 [00:03<00:
18, 1574.53it/s]	
18% [REDACTED]	6636/36051 [00:04<00:
18, 1612.89it/s]	
19% [REDACTED]	6798/36051 [00:04<00:
19, 1513.91it/s]	
19% [REDACTED]	6962/36051 [00:04<00:
18, 1546.23it/s]	
20% [REDACTED]	7161/36051 [00:04<00:
17, 1655.34it/s]	
20% [REDACTED]	7330/36051 [00:04<00:
18, 1569.00it/s]	
21% [REDACTED]	7491/36051 [00:04<00:
18, 1574.97it/s]	
21% [REDACTED]	7685/36051 [00:04<00:
17, 1660.86it/s]	
22% [REDACTED]	7884/36051 [00:04<00:
16, 1743.04it/s]	
22% [REDACTED]	8062/36051 [00:04<00:
16, 1710.43it/s]	
23% [REDACTED]	8253/36051 [00:04<00:
15, 1764.20it/s]	
23% [REDACTED]	8432/36051 [00:05<00:
16, 1707.02it/s]	
24% [REDACTED]	8605/36051 [00:05<00:
16, 1706.01it/s]	
24% [REDACTED]	8778/36051 [00:05<00:
15, 1707.33it/s]	
25% [REDACTED]	8953/36051 [00:05<00:
15, 1703.70it/s]	
25% [REDACTED]	9126/36051 [00:05<00:
15, 1709.23it/s]	
26% [REDACTED]	9324/36051 [00:05<00:
15, 1764.22it/s]	
26% [REDACTED]	9502/36051 [00:05<00:
15, 1700.42it/s]	
27% [REDACTED]	9674/36051 [00:05<00:
15, 1699.82it/s]	
27% [REDACTED]	9861/36051 [00:05<00:
15, 1744.16it/s]	
28% [REDACTED]	10037/36051 [00:06<00:
15, 1680.88it/s]	
28% [REDACTED]	10219/36051 [00:06<00:
15, 1720.15it/s]	
29% [REDACTED]	10411/36051 [00:06<00:
14, 1771.71it/s]	
29% [REDACTED]	10590/36051 [00:06<00:
14, 1704.97it/s]	
30% [REDACTED]	10766/36051 [00:06<00:

30%		10700/36051	[00:00<00:
14, 1708.51it/s]			
30%		10952/36051	[00:06<00:
14, 1740.19it/s]			
31%		11127/36051	[00:06<00:
14, 1732.57it/s]			
31%		11301/36051	[00:06<00:
14, 1673.17it/s]			
32%		11481/36051	[00:06<00:
14, 1702.59it/s]			
32%		11666/36051	[00:06<00:
14, 1730.90it/s]			
33%		11846/36051	[00:07<00:
13, 1737.88it/s]			
33%		12041/36051	[00:07<00:
13, 1779.55it/s]			
34%		12220/36051	[00:07<00:
13, 1741.45it/s]			
34%		12395/36051	[00:07<00:
14, 1617.82it/s]			
35%		12572/36051	[00:07<00:
14, 1655.60it/s]			
35%		12761/36051	[00:07<00:
13, 1701.57it/s]			
36%		12933/36051	[00:07<00:
13, 1668.19it/s]			
36%		13102/36051	[00:07<00:
13, 1653.16it/s]			
37%		13284/36051	[00:07<00:
13, 1698.28it/s]			
37%		13460/36051	[00:08<00:
13, 1711.71it/s]			
38%		13632/36051	[00:08<00:
13, 1633.51it/s]			
38%		13818/36051	[00:08<00:
13, 1691.80it/s]			
39%		14013/36051	[00:08<00:
12, 1746.09it/s]			
39%		14190/36051	[00:08<00:
12, 1720.45it/s]			
40%		14391/36051	[00:08<00:
12, 1788.59it/s]			
40%		14572/36051	[00:08<00:
12, 1771.60it/s]			
41%		14751/36051	[00:08<00:
12, 1735.59it/s]			
41%		14926/36051	[00:08<00:
12, 1722.30it/s]			
42%		15099/36051	[00:08<00:
12, 1696.78it/s]			
42%		15291/36051	[00:09<00:
11, 1739.55it/s]			
43%		15471/36051	[00:09<00:
11, 1754.81it/s]			
43%		15668/36051	[00:09<00:
11, 1798.44it/s]			
44%		15857/36051	[00:09<00:
11, 1823.70it/s]			
44%		16040/36051	[00:09<00:
11, 1813.66it/s]			
45%		16222/36051	[00:09<00:
10, 1811.70it/s]			
46%		16404/36051	[00:09<00:
10, 1812.39it/s]			
46%		16586/36051	[00:09<00:
11, 1756.92it/s]			
46%		16763/36051	[00:09<00:
11, 1741.90it/s]			
47%		16938/36051	[00:10<00:
11, 1710.76it/s]			
47%		17110/36051	[00:10<00:
11, 1708.65it/s]			
48%		17282/36051	[00:10<00:
11, 1613.36it/s]			
48%		17454/36051	[00:10<00:
11, 1639.54it/s]			
49%		17619/36051	[00:10<00:
11, 1570.27it/s]			

11, 1579.37it/s]		17798/36051 [00:10<00:
49%		
11, 1631.67it/s]		17993/36051 [00:10<00:
50%		
10, 1703.00it/s]		18165/36051 [00:10<00:
50%		
10, 1661.12it/s]		18333/36051 [00:10<00:
51%		
10, 1655.85it/s]		18514/36051 [00:10<00:
51%		
10, 1687.20it/s]		18693/36051 [00:11<00:
52%		
10, 1712.88it/s]		18865/36051 [00:11<00:
52%		
10, 1664.36it/s]		19033/36051 [00:11<00:
53%		
10, 1564.34it/s]		19205/36051 [00:11<00:
53%		
10, 1594.30it/s]		19388/36051 [00:11<00:
54%		
10, 1657.20it/s]		19571/36051 [00:11<00:
54%		
09, 1703.51it/s]		19746/36051 [00:11<00:
55%		
09, 1706.20it/s]		19924/36051 [00:11<00:
55%		
09, 1720.38it/s]		20100/36051 [00:11<00:
56%		
09, 1731.00it/s]		20274/36051 [00:12<00:
56%		
09, 1722.67it/s]		20464/36051 [00:12<00:
57%		
08, 1765.09it/s]		20651/36051 [00:12<00:
57%		
08, 1788.02it/s]		20831/36051 [00:12<00:
58%		
08, 1784.41it/s]		21021/36051 [00:12<00:
58%		
08, 1812.30it/s]		21203/36051 [00:12<00:
59%		
08, 1733.85it/s]		21378/36051 [00:12<00:
59%		
08, 1738.22it/s]		21553/36051 [00:12<00:
60%		
08, 1697.88it/s]		21728/36051 [00:12<00:
60%		
08, 1707.56it/s]		21900/36051 [00:12<00:
61%		
08, 1659.85it/s]		22067/36051 [00:13<00:
61%		
08, 1608.79it/s]		22234/36051 [00:13<00:
62%		
08, 1622.37it/s]		22424/36051 [00:13<00:
62%		
08, 1691.14it/s]		22604/36051 [00:13<00:
63%		
07, 1708.49it/s]		22776/36051 [00:13<00:
63%		
07, 1668.63it/s]		22944/36051 [00:13<00:
64%		
08, 1629.99it/s]		23126/36051 [00:13<00:
64%		
07, 1678.75it/s]		23295/36051 [00:13<00:
65%		
07, 1676.68it/s]		23464/36051 [00:13<00:
65%		
07, 1655.29it/s]		23640/36051 [00:13<00:
66%		
07, 1677.88it/s]		23809/36051 [00:14<00:
66%		
07, 1643.09it/s]		23974/36051 [00:14<00:
67%		
07, 1622.28it/s]		24148/36051 [00:14<00:
67%		
07, 1637.87it/s]		24313/36051 [00:14<00:
67%		
07, 1629.25it/s]		24488/36051 [00:14<00:
68%		

68%		24488/36051	[00:14<00:
06, 1660.88it/s]			
68%		24664/36051	[00:14<00:
06, 1688.78it/s]			
69%		24834/36051	[00:14<00:
06, 1658.07it/s]			
69%		25016/36051	[00:14<00:
06, 1690.29it/s]			
70%		25186/36051	[00:14<00:
06, 1649.13it/s]			
70%		25352/36051	[00:15<00:
06, 1612.75it/s]			
71%		25524/36051	[00:15<00:
06, 1639.63it/s]			
71%		25712/36051	[00:15<00:
06, 1704.55it/s]			
72%		25893/36051	[00:15<00:
05, 1719.98it/s]			
72%		26084/36051	[00:15<00:
05, 1767.08it/s]			
73%		26272/36051	[00:15<00:
05, 1791.56it/s]			
73%		26452/36051	[00:15<00:
05, 1792.50it/s]			
74%		26632/36051	[00:15<00:
05, 1774.67it/s]			
74%		26810/36051	[00:15<00:
05, 1690.25it/s]			
75%		26986/36051	[00:15<00:
05, 1700.89it/s]			
75%		27166/36051	[00:16<00:
05, 1722.34it/s]			
76%		27350/36051	[00:16<00:
05, 1739.72it/s]			
76%		27525/36051	[00:16<00:
04, 1722.57it/s]			
77%		27707/36051	[00:16<00:
04, 1743.65it/s]			
77%		27882/36051	[00:16<00:
04, 1695.49it/s]			
78%		28053/36051	[00:16<00:
04, 1697.45it/s]			
78%		28230/36051	[00:16<00:
04, 1713.75it/s]			
79%		28408/36051	[00:16<00:
04, 1732.93it/s]			
79%		28592/36051	[00:16<00:
04, 1761.28it/s]			
80%		28772/36051	[00:16<00:
04, 1753.40it/s]			
80%		28950/36051	[00:17<00:
04, 1749.38it/s]			
81%		29126/36051	[00:17<00:
04, 1714.82it/s]			
81%		29298/36051	[00:17<00:
03, 1704.83it/s]			
82%		29479/36051	[00:17<00:
03, 1731.89it/s]			
82%		29653/36051	[00:17<00:
03, 1641.80it/s]			
83%		29819/36051	[00:17<00:
03, 1638.43it/s]			
83%		29986/36051	[00:17<00:
03, 1646.97it/s]			
84%		30152/36051	[00:17<00:
03, 1633.08it/s]			
84%		30339/36051	[00:17<00:
03, 1694.08it/s]			
85%		30522/36051	[00:18<00:
03, 1731.81it/s]			
85%		30697/36051	[00:18<00:
03, 1712.27it/s]			
86%		30871/36051	[00:18<00:
03, 1719.94it/s]			
86%		31044/36051	[00:18<00:
03, 1652.72it/s]			
87%		31215/36051	[00:18<00:
03, 1667.45it/s]			

02, 1667.45it/s]		
87%		31383/36051 [00:18<00:
02, 1653.34it/s]		
88%		31566/36051 [00:18<00:
02, 1702.17it/s]		
88%		31737/36051 [00:18<00:
02, 1704.42it/s]		
89%		31908/36051 [00:18<00:
02, 1692.68it/s]		
89%		32094/36051 [00:18<00:
02, 1723.93it/s]		
90%		32273/36051 [00:19<00:
02, 1742.03it/s]		
90%		32466/36051 [00:19<00:
02, 1781.76it/s]		
91%		32645/36051 [00:19<00:
01, 1779.41it/s]		
91%		32824/36051 [00:19<00:
01, 1738.06it/s]		
92%		32999/36051 [00:19<00:
01, 1689.52it/s]		
92%		33172/36051 [00:19<00:
01, 1694.76it/s]		
93%		33350/36051 [00:19<00:
01, 1709.54it/s]		
93%		33522/36051 [00:19<00:
01, 1696.39it/s]		
93%		33692/36051 [00:19<00:
01, 1667.50it/s]		
94%		33860/36051 [00:20<00:
01, 1648.61it/s]		
94%		34026/36051 [00:20<00:
01, 1607.96it/s]		
95%		34215/36051 [00:20<00:
01, 1675.45it/s]		
95%		34389/36051 [00:20<00:
00, 1684.80it/s]		
96%		34559/36051 [00:20<00:
00, 1651.50it/s]		
96%		34725/36051 [00:20<00:
00, 1534.72it/s]		
97%		34886/36051 [00:20<00:
00, 1555.96it/s]		
97%		35044/36051 [00:20<00:
00, 1512.56it/s]		
98%		35197/36051 [00:20<00:
00, 1509.49it/s]		
98%		35386/36051 [00:20<00:
00, 1602.91it/s]		
99%		35549/36051 [00:21<00:
00, 1605.78it/s]		
99%		35712/36051 [00:21<00:
00, 1606.61it/s]		
100%		35889/36051 [00:21<00:
00, 1635.25it/s]		
100%		36051/36051 [00:21<00:
00, 1687.58it/s]		

In [313]:

```
tfidf2vec_title_train=tfidf2vec(x_train['title'].values)
tfidf2vec_title_cv=tfidf2vec(x_cv['title'].values)
tfidf2vec_title_test=tfidf2vec(x_test['title'].values)
```

0%		0/49039
[00:00<?, ?it/s]		
4%		1749/49039 [00:00<00:0
2, 17466.32it/s]		
10%		4696/49039 [00:00<00:0
2, 19748.72it/s]		
16%		7774/49039 [00:00<00:0
1, 22011.41it/s]		
23%		11040/49039 [00:00<00:0
1, 24374.94it/s]		
29%		14217/49039 [00:00<00:0
1, 26152.23it/s]		

```

1, 28028.56it/s] | 17714/49039 [00:00<00:0
43% | 21196/49039 [00:00<00:0
0, 29549.74it/s] | 24294/49039 [00:00<00:0
50% | 27539/49039 [00:00<00:0
0, 29795.71it/s] | 30572/49039 [00:01<00:0
62% | 33397/49039 [00:01<00:0
0, 30364.83it/s] | 36085/49039 [00:01<00:0
68% | 38931/49039 [00:01<00:0
0, 25022.72it/s] | 42735/49039 [00:01<00:0
74% | 45740/49039 [00:01<00:0
0, 25980.09it/s] | 48724/49039 [00:01<00:0
79% | 49039/49039 [00:01<00:0
0, 26616.15it/s] | 0/24155
87% | 2416/24155 [00:00<00:0
0, 29000.69it/s] | 5260/24155 [00:00<00:0
93% | 8441/24155 [00:00<00:0
0, 29103.25it/s] | 11462/24155 [00:00<00:0
99% | 14519/24155 [00:00<00:0
0, 29296.18it/s] | 17711/24155 [00:00<00:0
100% | 20655/24155 [00:00<00:0
0, 28489.85it/s] | 24044/24155 [00:00<00:0
0% | 24155/24155 [00:00<00:0
[00:00<?, ?it/s] | 0/36051
10% | 2459/36051 [00:00<00:0
0, 24006.35it/s] | 5302/36051 [00:00<00:0
22% | 8545/36051 [00:00<00:0
0, 25122.71it/s] | 11288/36051 [00:00<00:0
35% | 14474/36051 [00:00<00:0
0, 26710.83it/s] | 17534/36051 [00:00<00:0
47% | 20976/36051 [00:00<00:0
0, 27545.18it/s] | 24314/36051 [00:00<00:0
60% | 27802/36051 [00:00<00:0
0, 28126.74it/s] | 30891/36051 [00:01<00:0
73% | 33988/36051 [00:01<00:0
0, 29065.62it/s] | 36051/36051 [00:01<00:0
86% | 30373.31it/s]
100%

```

In [315]:

```
from scipy.sparse import hstack
```

```

x_tr_tfidf2vec=hstack((tfidf2vec_essays_train,tfidf2vec_title_train,x_train_clean_subcategories_oh,
x_train_clean_categories_oh,x_train_project_grade_category_oh,x_train_teacher_oh,x_train_state_oh,x
_train_prev_subj_stand,x_train QUAN_stand,x_train_price_stand)).tocsr()
x_te_tfidf2vec=hstack((tfidf2vec_essays_test,tfidf2vec_title_test,x_test_clean_subcategories_oh,x_t
est_clean_categories_oh,x_test_project_grade_category_oh,x_test_teacher_oh,x_test_state_oh,x_test_p
rev_subj_stand,x_test QUAN_stand,x_test_price_stand)).tocsr()
x_cv_tfidf2vec=hstack((tfidf2vec_essays_cv,tfidf2vec_title_cv,x_cv_clean_subcategories_oh,x_cv_clea
n_categories_oh,x_cv_project_grade_category_oh,x_cv_teacher_oh,x_cv_state_oh,x_cv_prev_subj_stand,x
_cv QUAN_stand,x_cv_price_stand)).tocsr()

print("Final Data matrix")
print(x_tr_tfidf2vec.shape, y_train.shape)
print(x_cv_tfidf2vec.shape, y_cv.shape)
print(x_te_tfidf2vec.shape, y_test.shape)
print("=="*100)

```

```

Final Data matrix
(49039, 702) (49039,)
(24155, 702) (24155,)
(36051, 702) (36051,)
=====

```

In [316]:

```

#Finding the Best possible value of Hyperparameter(HyperParameter tuning)

from sklearn.metrics import roc_auc_score
train_auc_tfidf_word=[]
cv_auc_tfidf_word=[]
c_list=[10**(-4),10**(-3),10**(-2),10**(-1),0.5,10**(0),10**(1),10**(2),10**(3),10**(4)]
for i in c_list:
    clf=LogisticRegression(C=i,class_weight='balanced')
    clf.fit(x_tr_tfidf2vec,y_train)
    y_train_pred=clf.predict_proba(x_tr_tfidf2vec)[:,-1]
    y_cv_pred=clf.predict_proba(x_cv_tfidf2vec)[:,-1]
    train_auc_tfidf_word.append(roc_auc_score(y_train,y_train_pred))
    cv_auc_tfidf_word.append(roc_auc_score(y_cv,y_cv_pred))

```

In [551]:

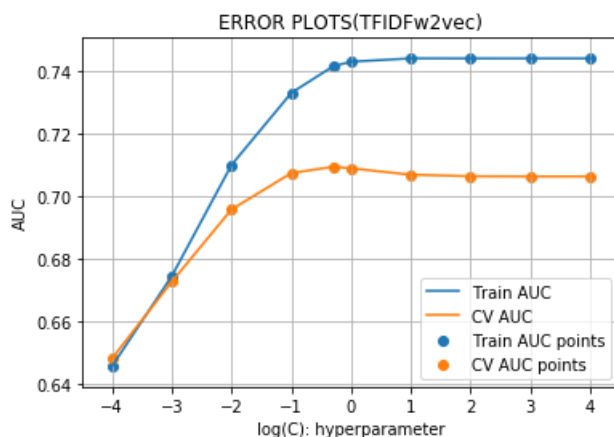
```

plt.plot(np.log10(c_list), train_auc_tfidf_word, label='Train AUC')
plt.plot(np.log10(c_list), cv_auc_tfidf_word, label='CV AUC')

plt.scatter(np.log10(c_list), train_auc_tfidf_word, label='Train AUC points')
plt.scatter(np.log10(c_list), cv_auc_tfidf_word, label='CV AUC points')

plt.legend()
plt.xlabel("log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS (TFIDFw2vec) ")
plt.grid()
plt.show()

```



In [320]:

```
best_c=0.5
```

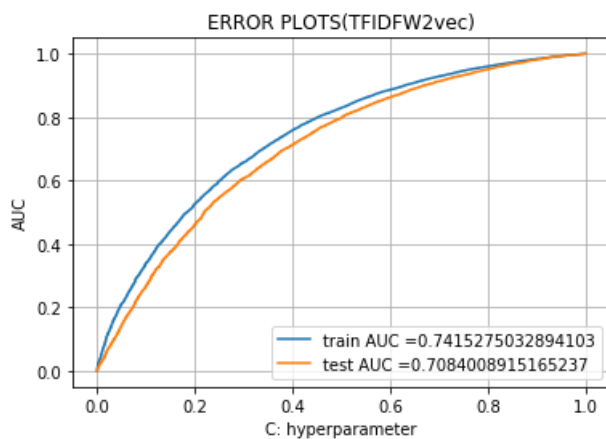
```
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression
```

```
clf_tfidf_word=LogisticRegression(C=best_c, class_weight='balanced')
clf_tfidf_word.fit(x_tr_tfidfw2vec, y_train)
```

```
y_train_pred_tfidf_word=clf_tfidf_word.predict_proba(x_tr_tfidfw2vec)[:,1]
y_test_pred_tfidf_word=clf_tfidf_word.predict_proba(x_te_tfidfw2vec)[:,1]
```

```
train_fpr_tfidf_word, train_tpr_tfidf_word, tr_thresholds_tfidf_word = roc_curve(y_train, y_train_pred_tfidf_word)
test_fpr_tfidf_word, test_tpr_tfidf_word, te_thresholds_tfidf_word = roc_curve(y_test, y_test_pred_tfidf_word)
```

```
plt.plot(train_fpr_tfidf_word, train_tpr_tfidf_word, label="train AUC =" + str(auc(train_fpr_tfidf_word, train_tpr_tfidf_word)))
plt.plot(test_fpr_tfidf_word, test_tpr_tfidf_word, label="test AUC =" + str(auc(test_fpr_tfidf_word, test_tpr_tfidf_word)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS (TFIDFW2vec)")
plt.grid()
plt.show()
```



In [554]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm_train_tfidf_word=confusion_matrix(y_train, predict(y_train_pred_tfidf_word, tr_thresholds_tfidf_word), train_fpr_tfidf_word, train_fpr_tfidf_word)
print("Test confusion matrix")
cm_test_tfidf_word=confusion_matrix(y_test, predict(y_test_pred_tfidf_word, tr_thresholds_tfidf_word), test_fpr_tfidf_word, test_fpr_tfidf_word)
```

Train confusion matrix

the maximum value of $tpr \cdot (1-fpr)$ 0.24999999546531537 for threshold 0.4

Test confusion matrix

the maximum value of $tpr \cdot (1-fpr)$ 0.24999999161092998 for threshold 0.472

In [555]:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
ax= plt.subplot()
sns.heatmap(cm_train_tfidf_word, annot=True, ax = ax, fmt='g'); #annot=True to annotate cells
```

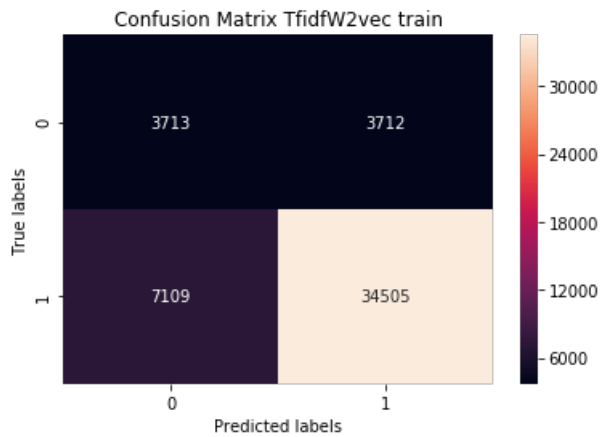
```
# labels, title and ticks
```

```
print(y_train.value_counts())
```

```
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
```

```
ax.set_title('Confusion Matrix TfidfW2vec train');
```

```
1    41614
0     7425
Name: project_is_approved, dtype: int64
```



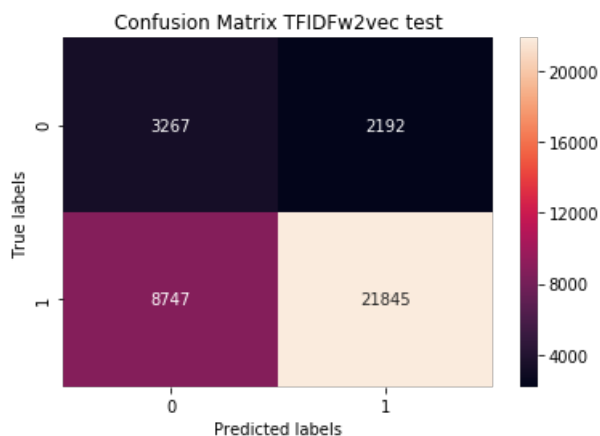
In [556]:

```
import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(cm_test_tfidf_word, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
print(y_test.value_counts())
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix TFIDFW2vec test');
```

```
1    30592
0     5459
Name: project_is_approved, dtype: int64
```



In [331]:

```
def count_words(essays):

    total_essay=[]
    for i in essays:
        count=0
        for word in i:
            count=count+1
        total_essay.append(count)
    return total_essay
```

In [446]:

```
#Word Count for each essay
essay words train=count words(x train['essay'].values)
```

```
essay_words_cv=count_words(x_cv['essay'].values)
essay_words_test=count_words(x_test['essay'].values)
print(len(essay_words_test))
```

36051

In [437]:

```
#Word Count for each title
title_words_train=count_words(x_train['title'].values)
title_words_cv=count_words(x_cv['title'].values)
title_words_test=count_words(x_test['title'].values)
```

In [343]:

```
#SentimentIntensityAnalyzer for each essay
```

```
def getsentiment(essays):

    analyser = SentimentIntensityAnalyzer()
    sentiment_list=[]
    for essay in essays:
        sent=analyser.polarity_scores(essay)
        sentiment_list.append(sent)
    return sentiment_list
```

In [345]:

```
sentiment_essay_train=getsentiment(x_train['essay'].values)
sentiment_essay_cv=getsentiment(x_cv['essay'].values)
sentiment_essay_test=getsentiment(x_test['essay'].values)
```

In [438]:

```
def getsentiments(sentiment_essay):
    pos_train=[]
    neg_train=[]
    compound_train=[]
    neu_train=[]
    for item in sentiment_essay:
        neg_train.append(item['neg'])
        pos_train.append(item['pos'])
        compound_train.append(item['compound'])
        neu_train.append(item['neu'])
    return neg_train,pos_train,compound_train,neu_train
```

In [439]:

```
neg_train,pos_train,compound_train,neu_train=getsentiments(sentiment_essay_train)
neg_cv,pos_cv ,compound_cv,neu_cv=getsentiments(sentiment_essay_cv)
```

In [478]:

```
neg_test,pos_test, compound_test, neu_test=getsentiments(sentiment_essay_test)
```

In [440]:

```
print(x_train_clean_categories_ohe.shape)
print(x_train_clean_subcategories_ohe.shape)
print(x_train_project_grade_category_ohe.shape)
```

```
(49039, 9)
(49039, 30)
(49039, 4)
```

In [441]:

```
print(x_train_teacher_ohc.shape)
print(x_train_state_ohc.shape)
print(x_train_prev_subj_stand.shape)
print(x_train_quan_stand.shape)
```

```
(49039, 5)
(49039, 51)
(49039, 1)
(49039, 1)
```

In [494]:

```
def tranpose(essay_words_train):
    essay_words_train=np.array(essay_words_train).reshape(49039,1)
    return np.array(essay_words_train)
```

In [495]:

```
def tranpose_test(essay_words_test_1):
    essay_words_test_1=(np.array(essay_words_test_1)).reshape(36051,1)
    return np.array(essay_words_test)
```

In [496]:

```
def tranpose_cv(essay_words_cv):
    essay_words_cv=np.array(essay_words_cv).reshape(24155,1)
    return np.array(essay_words_cv)
```

In [499]:

```
essay_words_train=tranpose(essay_words_train)
essay_words_test=tranpose_test(essay_words_test)
essay_words_cv=tranpose_cv(essay_words_cv)
```

In [500]:

```
title_words_train=tranpose(title_words_train)
title_words_test=tranpose_test(title_words_test)
title_words_cv=tranpose_cv(title_words_cv)
```

In [501]:

```
pos_train=tranpose(pos_train)
pos_test=tranpose_test(pos_test)
pos_cv=tranpose_cv(pos_cv)
```

In [502]:

```
neg_train=tranpose(neg_train)
neg_test=tranpose_test(neg_test)
neg_cv=tranpose_cv(neg_cv)
```

In [503]:

```
compound_train=tranpose(compound_train)
compound_test=tranpose_test(compound_test)
compound_cv=tranpose_cv(compound_cv)
```

In [523]:

```
neu_train=tranpose(neu_train)
neu_test1=tranpose_test(neu_test)
neu_cv=tranpose_cv(neu_cv)
```


In [528]:

```
essay1_words_test=essay_words_test
neu_test1=neu_test.reshape(36051,1)
pos_test1=pos_test.reshape(36051,1)
neg_test1=neg_test.reshape(36051,1)
compound_test1=compound_test.reshape(36051,1)
title_words_test1=title_words_test.reshape(36051,1)
essay_words_test1=essay_words_test.reshape(36051,1)
```

2.5 Logistic Regression with added Features `Set 5`

In [530]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
# a. Title, that describes your plot, this will be very helpful to the reader
# b. Legends if needed
# c. X-axis label
# d. Y-axis label

from scipy.sparse import hstack

print(essay_words_test.shape)
x_tr_5=hstack((x_train_clean_subcategories_ohc,x_train_clean_categories_ohc,x_train_project_grade_category_ohc,x_train_teacher_ohc,x_train_state_ohc,x_train_prev_subj_stand,x_train_quan_stand,x_train_price_stand,(essay_words_train),(title_words_train),(neg_train),(pos_train),(compound_train),(neu_train))).to_csr()
x_te_5=hstack((x_test_clean_subcategories_ohc,x_test_clean_categories_ohc,x_test_project_grade_category_ohc,x_test_teacher_ohc,x_test_state_ohc,x_test_prev_subj_stand,x_test_quan_stand,x_test_price_stand,neu_test1,pos_test1,neg_test1,compound_test1,essay_words_test1,title_words_test1)).to_csr()
x_cv_5=hstack((x_cv_clean_subcategories_ohc,x_cv_clean_categories_ohc,x_cv_project_grade_category_ohc,x_cv_teacher_ohc,x_cv_state_ohc,x_cv_prev_subj_stand,x_cv_quan_stand,x_cv_price_stand,(essay_words_cv),(title_words_cv),(neg_cv),(pos_cv),(compound_cv),(neu_cv))).to_csr()

print("Final Data matrix")
print(x_tr_5.shape, y_train.shape)
print(x_cv_5.shape, y_cv.shape)
print(x_te_5.shape, y_test.shape)
print("="*100)
```

```
(36051,)
Final Data matrix
(49039, 108) (49039,)
(24155, 108) (24155,)
(36051, 108) (36051,)
```

In [532]:

```
#Hyperparameter tuning for feature set 5(i.e without text data)

from sklearn.metrics import roc_auc_score
train_auc_5=[]
cv_auc_5=[]
c_list=[10**(-4),10**(-3),10**(-2),10**(-1),0.5,10**(0),10**(1),10**(2),10**(3),10**(4)]
for i in c_list:
    clf=LogisticRegression(C=i,class_weight='balanced')
    clf.fit(x_tr_5,y_train)
    y_train_pred_5=clf.predict_proba(x_tr_5)[:,-1]
    y_cv_pred_5=clf.predict_proba(x_cv_5)[:,-1]
    train_auc_5.append(roc_auc_score(y_train,y_train_pred_5))
    cv_auc_5.append(roc_auc_score(y_cv,y_cv_pred_5))
```

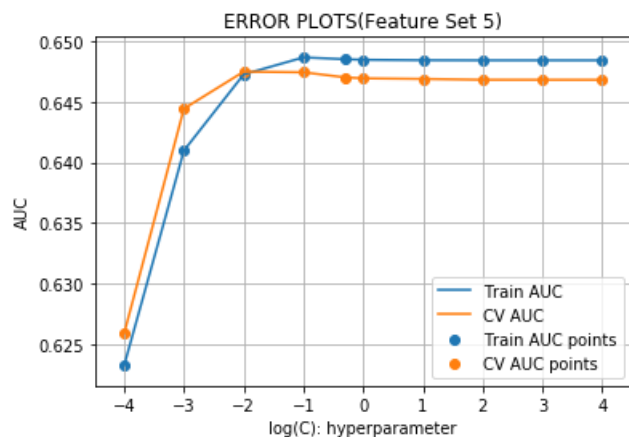
In [533]:

```
plt.plot(np.log10(c_list), train_auc_5, label='Train AUC')
```

```
plt.plot(np.log10(c_list), train_auc_5, label='Train AUC',
plt.plot(np.log10(c_list), cv_auc_5, label='CV AUC')

plt.scatter(np.log10(c_list), train_auc_5, label='Train AUC points')
plt.scatter(np.log10(c_list), cv_auc_5, label='CV AUC points')

plt.legend()
plt.xlabel("log(C): hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS (Feature Set 5)")
plt.grid()
plt.show()
```



In [534]:

```
best_c=10**-2

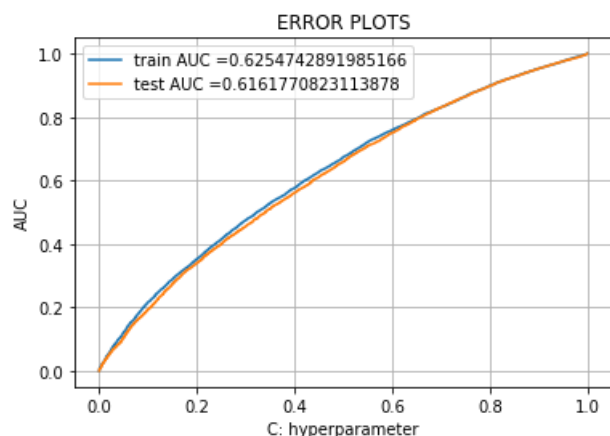
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression

clf_final_5=LogisticRegression(C=best_c, class_weight='balanced')
clf_final_5.fit(x_tr_bow, y_train)

y_train_pred_5=clf_final_5.predict_proba(x_tr_bow)[: ,1]
y_test_pred_5=clf_final_5.predict_proba(x_te_bow)[: ,1]

train_fpr_5, train_tpr_5, tr_thresholds_5 = roc_curve(y_train, y_train_pred_5)
test_fpr_5, test_tpr_5, te_thresholds_5 = roc_curve(y_test, y_test_pred_5)

plt.plot(train_fpr_5, train_tpr_5, label="train AUC =" +str(auc(train_fpr_5, train_tpr_5)))
plt.plot(test_fpr_5, test_tpr_5, label="test AUC =" +str(auc(test_fpr_5, test_tpr_5)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [559]:

```

from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
cm_train_5=confusion_matrix(y_train, predict(y_train_pred_5, tr_thresholds_5, train_fpr_5, train_fpr_5)
)
print("Test confusion matrix")
cm_test_5=confusion_matrix(y_test, predict(y_test_pred_5, tr_thresholds_5, test_fpr_5, test_fpr_5))

```

Train confusion matrix

the maximum value of $tpr \cdot (1-fpr)$ 0.24999999546531537 for threshold 0.483

Test confusion matrix

the maximum value of $tpr \cdot (1-fpr)$ 0.24999999161092998 for threshold 0.512

In [560]:

```

import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(cm_train_5, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

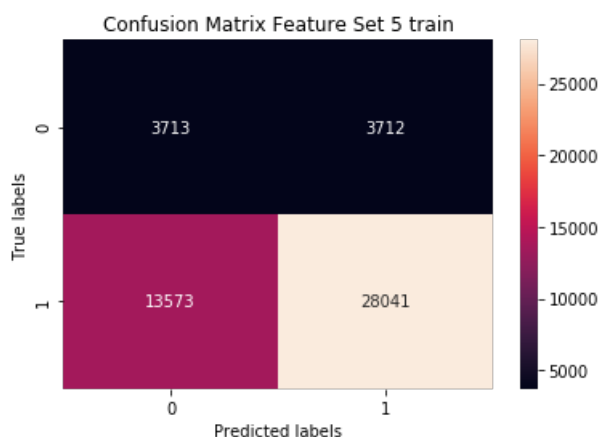
# labels, title and ticks
print(y_train.value_counts())
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix Feature Set 5 train');

```

```

1    41614
0     7425
Name: project_is_approved, dtype: int64

```



In [562]:

```

import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()
sns.heatmap(cm_test_5, annot=True, ax = ax,fmt='g'); #annot=True to annotate cells

# labels, title and ticks
print(y_test.value_counts())
ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix Feature Set 5 test');

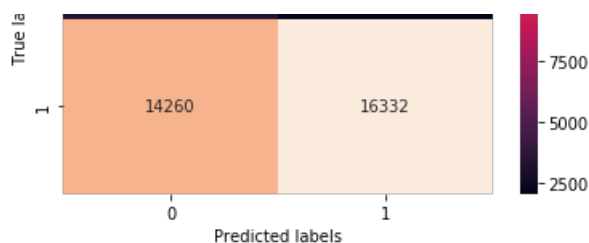
```

```

1    30592
0     5459
Name: project_is_approved, dtype: int64

```





3. Conclusion

In [1]:

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

# Please compare all your models using Prettytable library
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Model", "HyperParameter", "AUC(test)"]

x.add_row(["BOW", "LogisticRegression", 0.0001, 0.723])
x.add_row(["TFIDF", "Logistic Regression", 0.1, 0.719])
x.add_row(["AvgW2Vec", "Logistic Regression", 1, 0.70825])
x.add_row(["TFIDFW2vec", "Logistic Regression", 0.5, 0.7084])
x.add_row(["FeatureSet5(without text data)", "Logistic Regression", 0.01, 0.62])

print(x)
```

Vectorizer	Model	HyperParameter	AUC(test)
BOW	LogisticRegression	0.0001	0.723
TFIDF	Logistic Regression	0.1	0.719
AvgW2Vec	Logistic Regression	1	0.70825
TFIDFW2vec	Logistic Regression	0.5	0.7084
FeatureSet5(without text data)	Logistic Regression	0.01	0.62

1.As the AUC score was reduced in feature set 5 i.e when we did not take textual data,It seems like project_title and project_essays are one of the most important feature here and we cannot neglect such features while Modelling