

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>	<ul style="list-style-type: none">••	Title of the project. Examples: <code>Art Will Make You Happy!</code> <code>First Grade Fun</code>
<code>project_grade_category</code>	<ul style="list-style-type: none">••••	Grade level of students for which the project is targeted. One of the following enumerated values: <code>Grades PreK-2</code> <code>Grades 3-5</code> <code>Grades 6-8</code> <code>Grades 9-12</code>
<code>project_subject_categories</code>	<ul style="list-style-type: none">•••••••••	One or more (comma-separated) subject categories for the project from the following enumerated list of values: <code>Applied Learning</code> <code>Care & Hunger</code> <code>Health & Sports</code> <code>History & Civics</code> <code>Literacy & Language</code> <code>Math & Science</code> <code>Music & The Arts</code> <code>Special Needs</code> <code>Warmth</code> Examples: <ul style="list-style-type: none">• <code>Music & The Arts</code>• <code>Literacy & Language, Math & Science</code>
<code>school_state</code>		State where school is located (Two-letter U.S. postal code). Example: WY
<code>project_subject_subcategories</code>	<ul style="list-style-type: none">••	One or more (comma-separated) subject subcategories for the project. Examples: <code>Literacy</code> <code>Literature & Writing, Social Sciences</code>
<code>project_resource_summary</code>	<ul style="list-style-type: none">•	An explanation of the resources needed for the project. Example: <code>My students need hands on literacy materials to manage sensory needs!</code>
<code>project_essay_1</code>		First application essay*
<code>project_essay_2</code>		Second application essay*
<code>project_essay_3</code>		Third application essay*

Feature	Description
project_essay_4	Fourth application essay
project_submitted_datetime	Datetime when project application was submitted. Example: 2016-04-28 12:43:56.245
teacher_id	A unique identifier for the teacher of the proposed project. Example: bdf8baa8fedef6bfeec7ae4ff1c15c56
teacher_prefix	Teacher's title. One of the following enumerated values: <ul style="list-style-type: none"> nan Dr. Mr. Mrs. Ms. Teacher.
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same teacher. Example: 2

* See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
id	A <code>project_id</code> value from the <code>train.csv</code> file. Example: p036502
description	Description of the resource. Example: Tenor Saxophone Reeds, Box of 25
quantity	Quantity of the resource required. Example: 3
price	Price of the resource required. Example: 9.95

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in `train.csv`, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
project_is_approved	A binary flag indicating whether DonorsChoose approved the project. A value of 0 indicates the project was not approved, and a value of 1 indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- __project_essay_1__: "Introduce us to your classroom"
- __project_essay_2__: "Tell us more about your students"
- __project_essay_3__: "Describe how your students will use the materials you're requesting"
- __project_essay_3__: "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- __project_essay_1__: "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2__: "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with `project_submitted_datetime` of 2016-05-17 and later, the values of `project_essay_3` and `project_essay_4` will be NaN.

In [1]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
```

```

import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

```

1.1 Reading Data

In [2]:

```

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

```

In [3]:

```

print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)

```

Number of data points in train data (109248, 17)

```

-----
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
'project_submitted_datetime' 'project_grade_category'
'project_subject_categories' 'project_subject_subcategories'
'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
'project_essay_4' 'project_resource_summary'
'teacher_number_of_previously_posted_projects' 'project_is_approved']

```

In [4]:

```

print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)

```

Number of data points in train data (1541272, 4)

```
['id' 'description' 'quantity' 'price']
```

Out[4]:

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

1.2 preprocessing of project_subject_categories

In [5]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in 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 replacing 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 [6]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_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 replacing 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 [7]:

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

In [8]:

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

Out[8]:

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

In [9]:

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

In [10]:

```
# 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. \r\n\r\n We have over 24 languages represented in our English Learner program with students at every level of mastery. We also have over 40 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 at begs for more resources. Many times our parents are learning to read and speak English alongside of their children. Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English language even if no one 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.\r\n\r\nParents that do not have access to a dvd player will have the

develop early reading skills. All students 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.\r\nnnannan

=====

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 that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the 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\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\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.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an "open classroom" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing 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.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs 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 kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and in turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan

=====

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

about different levels and it is more accessible.nannan
=====

In [11]:

```
# 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 [12]:

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

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

In [13]:

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

In [14]:

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

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

In [15]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', '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', "d
esn'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 [16]:

```
# 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('\t', ' ')
    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())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248  
[00:59<00:00, 1826.92it/s]
```

In [17]:

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


'my kindergarten students varied disabilities ranging speech language delays cognitive delays gross fine motor delays autism they eager beavers always strive work hardest working past limitations the materials ones i seek students i teach title i school students receive free reduced price lunch despite disabilities limitations students love coming school come eager learn explore have ever felt like ants pants needed groove move meeting this kids feel time the want able move learn say wobble chairs answer i love develop core enhances gross motor turn fine motor skills they also want learn games kids not want sit worksheets they want learn count jumping playing physical engagement key success the number toss color shape mats make happen my students forget work fun 6 year old deserves nannan'

In [19]:

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248  
[00:03<00:00, 35727.25it/s]
```

```
# after preprocessing Project title
preprocessed_title[20000]
```

```
'we need to move it while we input it'
```

In [21]:

Out[21]:

we are going to consider

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

```

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

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

```

## 1.5.1 Vectorizing Categorical data

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

In [22]:

```

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 encoding ",categories_one_hot.shape)

```

```

['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encoding (109248, 9)

```

In [23]:

```

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 encoding ",sub_categories_one_hot.shape)

```

```

['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL',
'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encoding (109248, 30)

```

## 1.5.2 Vectorizing Text data

### 1.5.2.1 Bag of words

In [24]:

```

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 encoding ",text_bow.shape)

```

```

Shape of matrix after one hot encoding (109248, 16623)

```

In [25]:

```

Similarly you can vectorize for title also
Using above lines of code

vectorizerTitle = CountVectorizer(min_df=10)
text_bow_title = vectorizerTitle.fit_transform(preprocessed_title)
print("Shape of matrix after one hot encoding ",text_bow_title.shape)

```

```
print("Shape of matrix after one hot encoding ",text_body_title.shape)
```

Shape of matrix after one hot encoding (109248, 3329)

### 1.5.2.2 TFIDF vectorizer

In [27]:

```
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 encoding ",text_tfidf.shape)
```

Shape of matrix after one hot encoding (109248, 16623)

In [28]:

```
Similarly you can vectorize for title also
Using above lines of code

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

Shape of matrix after one hot encoding (109248, 3329)

### 1.5.2.3 Using Pretrained Models: Avg W2V

In [120]:

```
'''
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-and-load-variables-in-python/

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

'''

```

Out [120]:

```

'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef
loadGloveModel(gloveFile):\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 m
odel[word] = embedding\n print ("Done.",len(model)," words loaded!")\n return model\nmodel =
loadGloveModel(\glove.42B.300d.txt')\n\n# =====\nOutput:\n \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495 words loaded!\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))\n\nwords = set(words)\n\nprint("the unique words in the coupus",
len(words))\n\ninter_words = set(model.keys()).intersection(words)\n\nprint("The number of words tha
t are present in both glove vectors and our coupus", len(inter_words),
(",np.round(len(inter_words)/len(words)*100,3),"%")\n\nwords_courpus = {}\nwords_glove =
set(model.keys())\n\nfor i in words:\n if i in words_glove:\n words_courpus[i] = model[i]\r
print("word 2 vec length", len(words_courpus))\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 pic
kle\n\nwith open(\glove_vectors', \wb') as f:\n pickle.dump(words_courpus, f)\n\n\n'

```

In [121]:

```

stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-sa
ve-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 [122]:

```

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% | 109248/109248  
[00:42<00:00, 2578.04it/s]

109248  
300

In [123]:

```
100%|██| 109248/109248
[00:01<00:00, 60683.61it/s]
```

```
100%|██| 109248/109248
[06:20<00:00, 287.10it/s]
```

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

In [137]:

```
100%|██| 109248/109248
[00:05<00:00, 18873.74it/s]
```

### 1.5.3 Vectorizing Numerical features

```
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')
```

```
check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

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

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

In [39]:

\_\_\_\_\_

```
price_standardized
```

Out[39]:

```
array([[-0.3905327],
 [0.00239637],
 [0.59519138],
 ...,
 [-0.15825829],
 [-0.61243967],
 [-0.51216657]])
```

## 1.5.4 Merging all the above features

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

In [40]:

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

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

## Computing Sentiment Scores

In [42]:

```
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 w
ith the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multiple intelli
gences i use a wide range\
of techniques to help all my students succeed students in my class come from a variety of differen
t backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school is a carin
g community of successful \
learners which can be seen through collaborative student project based learning in and out of the
classroom kindergarteners \
in my class love to work with hands on materials and have many different opportunities to practice
a skill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspect of the ki
ndergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my students love to role pla
y in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try cooking with real food
i will take their idea \
and create common core cooking lessons where we learn important math and writing concepts while co
oking delicious healthy \
food for snack time my students will have a grounded appreciation for the work that went into maki
ng the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this project woul
d expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade apple
sauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create our own cook
books to be printed and \
shared with families students will gain math and literature skills as well as a life long enjoymen
t 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
```

```
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

## Assignment 5: Logistic Regression

### 1. [Task-1] Logistic Regression(either SGDClassifier with log loss, or LogisticRegression) on these feature sets

- **Set 1:** categorical, numerical features + project\_title(BOW) + preprocessed\_eassay ('BOW with bi-grams' with 'min\_df=10' and 'max\_features=5000')
- **Set 2:** categorical, numerical features + project\_title(TFIDF)+ preprocessed\_eassay ('TFIDF with bi-grams' with 'min\_df=10' and 'max\_features=5000')
- **Set 3:** categorical, numerical features + project\_title(AVG W2V)+ preprocessed\_eassay (AVG W2V)
- **Set 4:** categorical, numerical features + project\_title(TFIDF W2V)+ preprocessed\_eassay (TFIDF W2V)

### 2. Hyper paramter tuning (find best hyper parameters corresponding the algorithm that you choose)

- Find the best hyper parameter which will give the maximum [AUC](#) value
- Find the best hyper paramter using k-fold cross validation or simple cross validation data
- Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

### 3. Representation of results

- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure.
- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
- Along with plotting ROC curve, you need to print the [confusion matrix](#) with predicted and original labels of test data points. Please visualize your confusion matrices using [seaborn heatmaps](#).

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

#### 5. Consider these set of features **Set 5**:

- [school\\_state](#) : categorical data
- [clean\\_categories](#) : categorical data
- [clean\\_subcategories](#) : categorical data
- [project\\_grade\\_category](#) :categorical data
- [teacher\\_prefix](#) : categorical data
- [quantity](#) : numerical data
- [teacher\\_number\\_of\\_previously\\_posted\\_projects](#) : numerical data
- [price](#) : numerical data
- [sentiment\\_score's\\_of\\_each\\_of\\_the\\_essay](#) : numerical data
- [number\\_of\\_words\\_in\\_the\\_title](#) : numerical data
- [number\\_of\\_words\\_in\\_the\\_combine\\_essays](#) : numerical data

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

### 6. Conclusion

- You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this [prettytable library link](#)

### Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakage, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit\_transform() on you train data, and apply the method transform() on cv/test data.



4. For more details please go through this [link](#).

## 2. Logistic Regression

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

In [43]:

```
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

Data splitting
Using project_data, which is a merge of project_data , price_date tables

from sklearn.model_selection import train_test_split
Donor_train, Donor_test, Approved_train, Approved_test = train_test_split(project_data, project_data[
 'project_is_approved'], test_size=0.33, stratify=project_data['project_is_approved'])
print(Donor_train.shape, Approved_train.shape)
print(Donor_test.shape, Approved_test.shape)
project_data.columns

(73196, 20) (73196,)
(36052, 20) (36052,)
```

Out[43]:

```
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', 'price',
 'quantity'],
 dtype='object')
```

### 2.2 Make Data Model Ready: encoding numerical, categorical features

In [44]:

```
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
make sure you featurize train and test data separatly

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

One hot encoding the catogorical features : School State, Clean Categories, Clean Sub-Categories
, Project Grade and Teacher Prefix

One hot Encoding for School State
vectorizer = CountVectorizer()
vectorizer.fit(Donor_train['school_state'].values) # fit has to happen only on train data
Donor_train_state_ohe = vectorizer.transform(Donor_train['school_state'].values)
Donor_test_state_ohe = vectorizer.transform(Donor_test['school_state'].values)

Print One Hot Encoding - School State output
print("After vectorizations School state")
print(Donor_train_state_ohe.shape, Approved_train.shape)
print(Donor_test_state_ohe.shape, Approved_test.shape)
```

```
print(Donor_test_state_one.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

After vectorizations School state

```
(73196, 51) (73196,)
(36052, 51) (36052,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'k',
s', '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']
=====
```

In [45]:

```
Preprocessing Project grade
from collections import Counter
my_counter = Counter()
for word in project_data['project_grade_category'].values:
 my_counter.update(word.split(" ",1))

project_grade_category_dict = dict(my_counter)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items(), key=lambda
kv: kv[1]))
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.keys()), lowercase
=False, binary=True)

One Hot Encoding - Project grade category

vectorizer.fit(Donor_train['project_grade_category'].values) # fit has to happen only on train
data
Donor_train_grade_ohe = vectorizer.transform(Donor_train['project_grade_category'].values)
Donor_test_grade_ohe = vectorizer.transform(Donor_test['project_grade_category'].values)

Print One Hot Encoding - Project grade output
print("After vectorizations Project grade category")
print(Donor_train_grade_ohe.shape, Approved_train.shape)
print(Donor_test_grade_ohe.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

After vectorizations Project grade category

```
(73196, 4) (73196,)
(36052, 4) (36052,)
['Grades 9-12', 'Grades 6-8', 'Grades 3-5', 'Grades PreK-2']
=====
```

In [46]:

```
One hot Encoding for project subject categories
vectorizer = CountVectorizer()
vectorizer.fit(Donor_train['clean_categories'].values) # fit has to happen only on train data
Donor_train_clean_cat_ohe = vectorizer.transform(Donor_train['clean_categories'].values)
Donor_test_clean_cat_ohe = vectorizer.transform(Donor_test['clean_categories'].values)

Print One Hot Encoding - Project subject output
print("After vectorizations project subject categories")
print(Donor_train_clean_cat_ohe.shape, Approved_train.shape)
print(Donor_test_clean_cat_ohe.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

After vectorizations project subject categories

```
(73196, 9) (73196,)
(36052, 9) (36052,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language',
'math_science', 'music_arts', 'specialneeds', 'warmth']
=====
```

In [47]:

```
One hot Encoding for project subject subcategories
vectorizer = CountVectorizer()
vectorizer.fit(Donor_train['clean_subcategories'].values) # fit has to happen only on train data
Donor_train_clean_subcat_ohe = vectorizer.transform(Donor_train['clean_subcategories'].values)
Donor_test_clean_subcat_ohe = vectorizer.transform(Donor_test['clean_subcategories'].values)

Print One Hot Encoding - project subject subcategories output
print("After vectorizations project subject subcategories")
print(Donor_train_clean_subcat_ohe.shape, Approved_train.shape)
print(Donor_test_clean_subcat_ohe.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

After vectorizations project subject subcategories

```
(73196, 30) (73196,)
(36052, 30) (36052,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government',
'college_careerprep', 'communityservice', 'earlydevelopment', 'economics', 'environmentalscience',
'esl', 'extracurricular', 'financialliteracy', 'foreignlanguages', 'gym_fitness',
'health_lifescience', 'health_wellness', 'history_geography', 'literacy', 'literature_writing', 'mathematics', 'music', 'nutritioneducation', 'other', 'parentinvolvement', 'performingarts', 'socialsciences', 'specialneeds', 'teamsports', 'visualarts', 'warmth']
=====
```

In [48]:

```
To avoid np.NaN invalid document error;
Source : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np-nan-is-an-invalid-document/39308809
One hot Encoding for Teacher Prefix
vectorizer = CountVectorizer()
vectorizer.fit(Donor_train['teacher_prefix'].values.astype('U')) # fit has to happen only on train data
Donor_train_teacher_ohe = vectorizer.transform(Donor_train['teacher_prefix'].values.astype('U'))
Donor_test_teacher_ohe = vectorizer.transform(Donor_test['teacher_prefix'].values.astype('U'))

Print One Hot Encoding - Teacher Prefix output
print("After vectorizations Teacher Prefix")
print(Donor_train_teacher_ohe.shape, Approved_train.shape)
print(Donor_test_teacher_ohe.shape, Approved_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

After vectorizations Teacher Prefix

```
(73196, 6) (73196,)
(36052, 6) (36052,)
['dr', 'mr', 'mrs', 'ms', 'nan', 'teacher']
=====
```

## Numerical Features

In [49]:

```
Normalizing the Numerical data : Price
Using code from Sample solution

from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(Donor_train['price'].values.reshape(-1,1))
Donor_train_price_norm = normalizer.transform(Donor_train['price'].values.reshape(-1,1))
Donor_test_price_norm = normalizer.transform(Donor_test['price'].values.reshape(-1,1))

print("After vectorizations Numerical Data: Price")
print(Donor_train_price_norm.shape, Approved_train.shape)
print(Donor_test_price_norm.shape, Approved_test.shape)
print("="*100)
```

After vectorizations Numerical Data: Price

```
(73196, 1) (73196,)
(36052, 1) (36052,)
=====
```

In [50]:

```
Normalizing the Numerical data : teacher_number_of_previously_posted_projects

from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(Donor_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
Donor_train_postedCount_norm =
normalizer.transform(Donor_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
Donor_test_postedCount_norm =
normalizer.transform(Donor_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print("After vectorizations Numerical Data: Previously Posted Projects")
print(Donor_train_postedCount_norm.shape, Approved_train.shape)
print(Donor_test_postedCount_norm.shape, Approved_test.shape)
print("="*100)
```

```
After vectorizations Numerical Data: Previously Posted Projects
(73196, 1) (73196,)
(36052, 1) (36052,)
=====
```

In [51]:

```
Normalizing the Numerical data : Quantity
Using code from Sample solution

from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(Donor_train['quantity'].values.reshape(-1,1))
Donor_train_quantity_norm = normalizer.transform(Donor_train['quantity'].values.reshape(-1,1))
Donor_test_quantity_norm = normalizer.transform(Donor_test['quantity'].values.reshape(-1,1))

print("After vectorizations Numerical Data: Quantity")
print(Donor_train_quantity_norm.shape, Approved_train.shape)
print(Donor_test_quantity_norm.shape, Approved_test.shape)
print("="*100)
```

```
After vectorizations Numerical Data: Quantity
(73196, 1) (73196,)
(36052, 1) (36052,)
=====
```

## 2.3 Make Data Model Ready: encoding eassay, and project\_title

### BoW : Project Essays

In [53]:

```
Using sample solution code
Using max features as 5000 ,min df = 10 and bigrams

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,2), max_features=5000)
vectorizer.fit(Donor_train['essay'].values) # fit has to happen only on train data
Donor_train_essay_bow = vectorizer.transform(Donor_train['essay'].values)
Donor_test_essay_bow = vectorizer.transform(Donor_test['essay'].values)

print("After vectorizing Project Essays BoW")
print(Donor_train_essay_bow.shape, Approved_train.shape)
```

```
print(Donor_test_essay_bow.shape, Approved_test.shape)
print("="*100)
```

After vectorizing Project Essays BoW  
(73196, 5000) (73196,)  
(36052, 5000) (36052,)

=====



## BoW : Project Title

In [54]:

```
Using sample solution code
Using max features as 3000, to make column size same

from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10,ngram_range=(1,2), max_features=3000)
vectorizer.fit(Donor_train['project_title'].values) # fit has to happen only on train data
Donor_train_title_bow = vectorizer.transform(Donor_train['project_title'].values)
Donor_test_title_bow = vectorizer.transform(Donor_test['project_title'].values)

print("After vectorizing Project Essays BoW")
print(Donor_train_title_bow.shape, Approved_train.shape)
print(Donor_test_title_bow.shape, Approved_test.shape)
print("="*100)
```

After vectorizing Project Essays BoW  
(73196, 3000) (73196,)  
(36052, 3000) (36052,)

=====



## TFIDF : Project Essays

In [94]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,2), max_features=5000)
vectorizer.fit(Donor_train['essay'].values) # fit has to happen only on train data
Donor_train_essay_tfidf = vectorizer.transform(Donor_train['essay'].values)
Donor_test_essay_tfidf = vectorizer.transform(Donor_test['essay'].values)

print("After vectorizing Project Essays TFIDF")
print(Donor_train_essay_tfidf.shape, Approved_train.shape)
print(Donor_test_essay_tfidf.shape, Approved_test.shape)
print("="*100)
```

After vectorizing Project Essays TFIDF  
(73196, 5000) (73196,)  
(36052, 5000) (36052,)

=====



## TFIDF : Project Title

In [97]:

```
%%time
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10,ngram_range=(1,2), max_features=3000)
vectorizer.fit(Donor_train['project_title'].values) # fit has to happen only on train data
Donor_train_title_tfidf = vectorizer.transform(Donor_train['project_title'].values)
Donor_test_title_tfidf = vectorizer.transform(Donor_test['project_title'].values)

print("After vectorizing Project Title using TFIDF: ")
print(Donor_train_title_tfidf.shape, Approved_train.shape)
print(Donor_test_title_tfidf.shape, Approved_test.shape)
```

```
After vectorizing Project Title using TFIDF:
(73196, 3000) (73196,)
(36052, 3000) (36052,)
```

◀ ▶ 🔍 📄 📁 📂 📅 📆 📇 📈 📉 📊 📋 📌 📍 📎 📏 📐 📑 📒 📓 📔 📕 📖 📗 📘 📙 📚 📛 📜 📝 📞 📟 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿 📠 📡 📢 📣 📤 📥 📦 📧 📨 📩 📪 📫 📬 📭 📮 📯 📰 📱 📲 📳 📴 📵 📶 📷 📸 📹 📺 📻 📼 📽 📾 📿

In [124]:

```
100%|██| 73196/73196
[00:30<00:00, 2422.44it/s]
```

```
100%|██| 36052/36052
[00:14<00:00, 2458.39it/s]
```

In [125]:

```
AVG W2V for Train Data
avg_w2v_vectors_train_title = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(Donor_train['project_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:
```



```
100%|██| 73196/73196 [12
:17<00:00, 99.19it/s]
```

73196  
300

```
100%|██| 36052/36052 [05:
54<00:00, 101.67it/s]
```

36052  
300

In [141]:

```
100%|██| 73196/73196
[00:02<00:00, 34091.85it/s]
```

73196  
300

```
100%|██| 36052/36052
[00:00<00:00, 45992.36it/s]
```

36052  
300



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

Apply Logistic Regression on different kind of featurization as mentioned in the instructions  
For Every model that you work on make sure you do the step 2 and step 3 of instructions

### 2.4.1 Applying Logistic Regression on BOW, SET 1

Set 1: categorical, numerical features + project\_title(BOW) + preprocessed\_eassay ( BOW with bi-grams with min\_df=10 and max\_features=5000 )

In [55]:

```
Please write all the code with proper documentation
merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
Using sample solution code

from scipy.sparse import hstack
Donor_tr = hstack((Donor_train_essay_bow, Donor_train_title_bow, Donor_train_state_oh,
Donor_train_teacher_oh, Donor_train_grade_oh,
Donor_train_clean_cat_oh, Donor_train_clean_subcat_oh, Donor_train_price_norm, Donor_train_postedCount_norm, Donor_train_quantity_norm)).tocsr()
Donor_te = hstack((Donor_test_essay_bow, Donor_test_title_bow, Donor_test_state_oh,
Donor_test_teacher_oh, Donor_test_grade_oh, Donor_test_clean_cat_oh, Donor_test_clean_subcat_oh,
Donor_test_price_norm, Donor_test_postedCount_norm, Donor_test_quantity_norm)).tocsr()

print("Final Donor Data Matrix for Set 1")
print(Donor_tr.shape, Approved_train.shape)
print(Donor_te.shape, Approved_test.shape)
print("="*100)
```

```
Final Donor Data Matrix for Set 1
(73196, 8103) (73196,)
(36052, 8103) (36052,)
```

In [64]:

```
%%time
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

parameters = [{'C': [10**-4, 10**-2, 10**0, 10**2, 10**4]}]

#Using GridSearchCV
model = GridSearchCV(LogisticRegression(max_iter=1000), parameters, scoring = 'roc_auc', cv=5, return_train_score = True)
model.fit(Donor_tr, Approved_train)

print(model.best_estimator_)
print(model.score(Donor_te, Approved_test))
```

```
LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
 intercept_scaling=1, max_iter=1000, multi_class='warn',
 n_jobs=None, penalty='l2', random_state=None, solver='warn',
 tol=0.0001, verbose=0, warm_start=False)
```

0.7148416313365006

Wall time: 1h 34min 24s

In [72]:

```
Plotting AUC for Train and Test data of Set 1

hyperparams = [10**-4, 10**-2, 10**0, 10**2, 10**4]
```

```

train_auc= model.cv_results_['mean_train_score']
train_auc_std= model.cv_results_['std_train_score']
test_auc = model.cv_results_['mean_test_score']
test_auc_std= model.cv_results_['std_test_score']

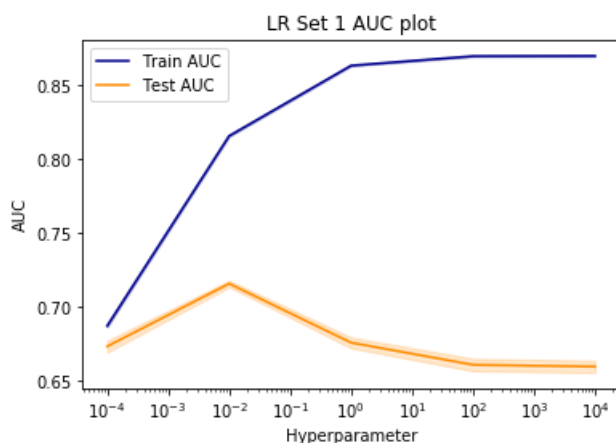
plt.figure()
plt.title('LR Set 1 AUC plot')
plt.xlabel('Hyperparameter')
plt.ylabel('AUC')

plt.semilogx(hyperparams, train_auc, label='Train AUC',color='darkblue')
plt.gca().fill_between(hyperparams,train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.semilogx(hyperparams, test_auc,label='Test AUC', color='darkorange')
plt.gca().fill_between(hyperparams, test_auc - test_auc_std, test_auc + test_auc_std,alpha=0.2,color='darkorange')

plt.legend(loc='best')
plt.show()

```



In [88]:

```

%%time
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

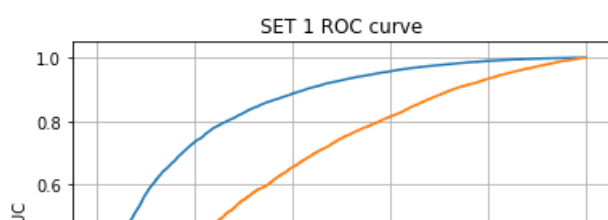
set1Model = LogisticRegression(C=100, penalty='l2',max_iter=500);
set1Model.fit(Donor_tr, Approved_train)

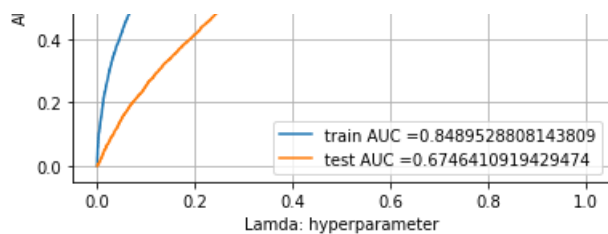
Approved_train_pred = set1Model.predict_proba(Donor_tr)
Approved_test_pred = set1Model.predict_proba(Donor_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(Approved_train, Approved_train_pred[:, 1])
test_fpr, test_tpr, te_thresholds = roc_curve(Approved_test, Approved_test_pred[:, 1])

plt.plot(train_fpr, train_tpr, label="train AUC =" +str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Lamda: hyperparameter")
plt.ylabel("AUC")
plt.title("SET 1 ROC curve")
plt.grid()
plt.show()

```





Wall time: 13min 18s

In [89]:

```
Writing own function for predict, with defined threshold
we will pick a threshold that will give the least fpr
def predictcm(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 [90]:

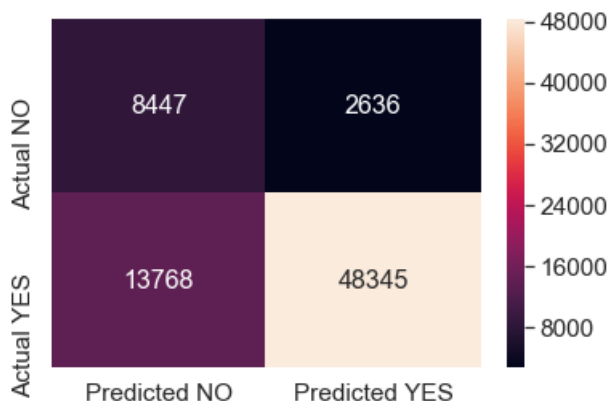
```
Confusion Matrix for Set 1 Train vs Test data

print("Train confusion matrix for Set 1")
donor_SET1_tr = pd.DataFrame(confusion_matrix(Approved_train, predictcm(Approved_train_pred[:, 1],
tr_thresholds, train_fpr, train_tpr)))
donor_SET1_tr.columns = ['Predicted NO', 'Predicted YES']
donor_SET1_tr = donor_SET1_tr.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4) #for label size
sns.heatmap(donor_SET1_tr, annot=True, annot_kws={"size": 16}, fmt='g')
```

Train confusion matrix for Set 1  
the maximum value of  $tpr*(1-fpr)$  0.5932178625888368 for threshold 0.828

Out[90]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c401b0bda0>



In [92]:

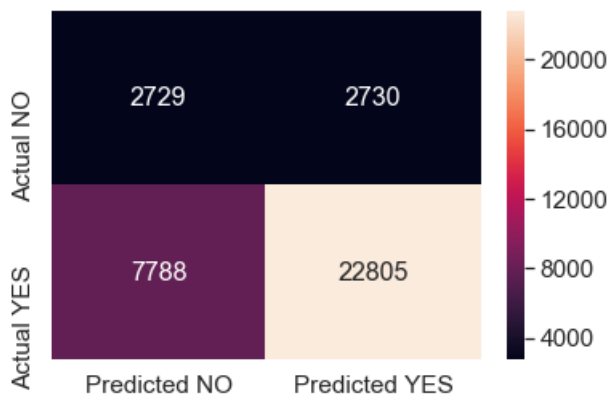
```
print("="*100)
print("Test confusion matrix for Set 1")
donor_SET1_te = pd.DataFrame(confusion_matrix(Approved_test, predict(Approved_test_pred[:, 1],
te_thresholds, test_fpr, test_tpr)))
```

```
donor_SET1_te.columns = ['Predicted NO', 'Predicted YES']
donor_SET1_te = donor_SET1_te.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)#for label size
sns.heatmap(donor_SET1_te, annot=True,annot_kws={"size": 16}, fmt='g')
```

Test confusion matrix for Set 1  
the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.818

Out[92]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c401b1d208>



## 2.4.2 Applying Logistic Regression on TFIDF, SET 2

Set 2: categorical, numerical features + project\_title(TFIDF)+ preprocessed\_eassay ( TFIDF with bi-grams with min\_df=10 and max\_features=5000 )

In [99]:

```
Using sample souldion code
from scipy.sparse import hstack
Donor_tr_tfidf = hstack((Donor_train_essay_tfidf, Donor_train_title_tfidf, Donor_train_state_ohe,
Donor_train_teacher_ohe, Donor_train_grade_ohe,
Donor_train_clean_cat_ohe, Donor_train_clean_subcat_ohe, Donor_train_price_norm, Donor_train_postedCount_norm, Donor_train_quantity_norm)).tocsr()
Donor_te_tfidf = hstack((Donor_test_essay_tfidf, Donor_test_title_tfidf, Donor_test_state_ohe,
Donor_test_teacher_ohe, Donor_test_grade_ohe, Donor_test_clean_cat_ohe, Donor_test_clean_subcat_ohe,
Donor_test_price_norm, Donor_test_postedCount_norm, Donor_test_quantity_norm)).tocsr()

print("Donor Data Matrix for Set 2")
print(Donor_tr_tfidf.shape, Approved_train.shape)
print(Donor_te_tfidf.shape, Approved_test.shape)
print("="*100)
```

Donor Data Matrix for Set 2  
(73196, 8103) (73196,)  
(36052, 8103) (36052,)

In [100]:

```
%time
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

parameters = [{'C': [10**-4, 10**-2, 10**0, 10**2, 10**4]}]

#Using GridSearchCV
model = GridSearchCV(LogisticRegression(max_iter=300), parameters, scoring = 'roc_auc', cv=5, return_train_score = True)
```

```
model.fit(Donor_tr_tfidf, Approved_train)

print(model.best_estimator_)
print(model.score(Donor_te_tfidf, Approved_test))
```

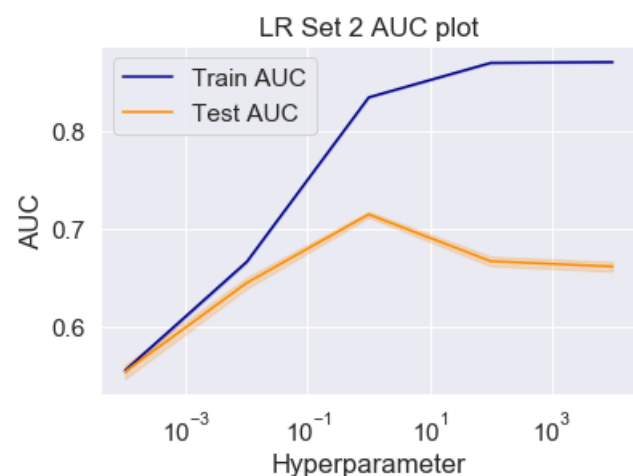
```
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
 intercept_scaling=1, max_iter=300, multi_class='warn',
 n_jobs=None, penalty='l2', random_state=None, solver='warn',
 tol=0.0001, verbose=0, warm_start=False)
0.7113057835049937
Wall time: 24min 58s
```

In [101]:

```
Plotting AUC for Train and Test data of Set 2
```

```
hyperparams = [10**-4, 10**-2, 10**0, 10**2, 10**4]
set2_train_auc= model.cv_results_['mean_train_score']
set2_train_auc_std= model.cv_results_['std_train_score']
set2_test_auc = model.cv_results_['mean_test_score']
set2_test_auc_std= model.cv_results_['std_test_score']

plt.figure()
plt.title('LR Set 2 AUC plot')
plt.xlabel('Hyperparameter')
plt.ylabel('AUC')
plt.semilogx(hyperparams, set2_train_auc, label='Train AUC',color='darkblue')
plt.gca().fill_between(hyperparams,set2_train_auc - set2_train_auc_std,set2_train_auc + set2_train_auc_std,alpha=0.2,color='darkblue')
plt.semilogx(hyperparams, set2_test_auc,label='Test AUC', color='darkorange')
plt.gca().fill_between(hyperparams, set2_test_auc - set2_test_auc_std, set2_test_auc + set2_test_auc_std,alpha=0.2,color='darkorange')
plt.legend(loc='best')
plt.show()
```



In [111]:

```
%%time
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

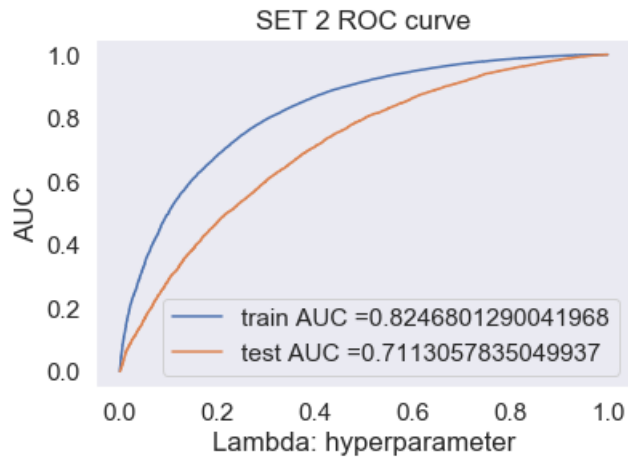
set2Model = LogisticRegression(C=1, penalty='l2',max_iter=500);
set2Model.fit(Donor_tr_tfidf, Approved_train)

Approved_train_pred_set2 = set2Model.predict_proba(Donor_tr_tfidf)
Approved_test_pred_set2 = set2Model.predict_proba(Donor_te_tfidf)

train_fpr, train_tpr, tr_thresholds = roc_curve(Approved_train, Approved_train_pred_set2[:, 1])
test_fpr, test_tpr, te_thresholds = roc_curve(Approved_test, Approved_test_pred_set2[:, 1])

plt.plot(train_fpr, train_tpr, label="train AUC =" +str(auc(train_fpr, train_tpr)))
```

```
plt.plot(test_fpr, test_tpr, label="test AUC "+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Lambda: hyperparameter")
plt.ylabel("AUC")
plt.title("SET 2 ROC curve")
plt.grid()
plt.show()
```



Wall time: 26.2 s

In [112]:

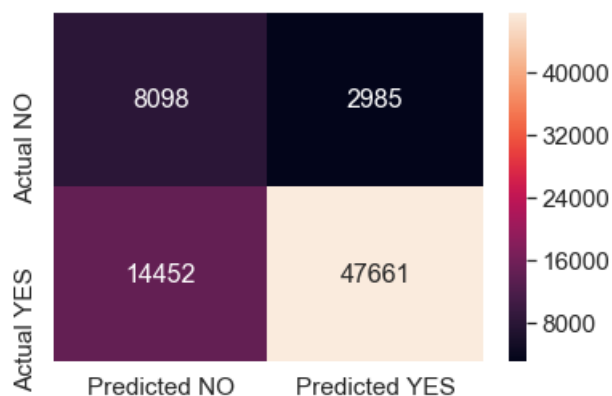
```
Confusion Matrix for Set 1 Train vs Test data

print("Train confusion matrix for Set 2")
donor_SET2_tr = pd.DataFrame(confusion_matrix(Approved_train,
predictcm(Approved_train_pred_set2[:, 1], tr_thresholds, train_fpr, train_tpr)))
donor_SET2_tr.columns = ['Predicted NO', 'Predicted YES']
donor_SET2_tr = donor_SET2_tr.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)#for label size
sns.heatmap(donor_SET2_tr, annot=True,annot_kws={"size": 16}, fmt='g')
```

Train confusion matrix for Set 2  
the maximum value of  $tpr \cdot (1 - fpr)$  0.5606619506580797 for threshold 0.823

Out[112]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c4019bc5c0>



In [113]:

```
print("="*100)
print("Test confusion matrix for Set 2")
donor_SET2_te = pd.DataFrame(confusion_matrix(Approved_test, predict(Approved_test_pred_set2[:,
1], te_thresholds, test_fpr, test_tpr)))
donor_SET2_te.columns = ['Predicted NO', 'Predicted YES']
donor_SET2_te = donor_SET2_te.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)#for label size
```

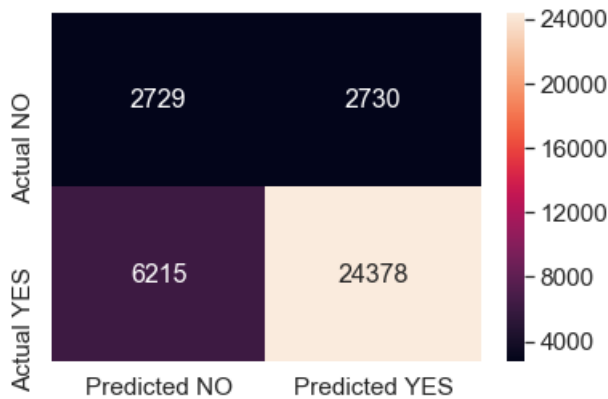
```
sns.heatmap(donor_SET2_te, annot=True,annot_kws={"size": 16}, fmt='g')
```

Test confusion matrix for Set 2

the maximum value of  $tpr \cdot (1 - fpr)$  0.249999999161092998 for threshold 0.794

Out[113]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c401a6cb38>



## 2.4.2 Applying Logistic Regression on AvgW2V, SET 3

Set 3: categorical, numerical features + project\_title(AVG W2V)+ preprocessed\_eassay (AVG W2V)

In [127]:

```
from scipy.sparse import hstack
Donor_tr_AvgW2V = hstack((avg_w2v_vectors_train_essays,avg_w2v_vectors_train_title,
Donor_train_state_ohe, Donor_train_teacher_ohe, Donor_train_grade_ohe, Donor_train_clean_cat_ohe,D
onor_train_clean_subcat_ohe,Donor_train_price_norm,Donor_train_postedCount_norm,Donor_train_quantit
y_norm)).tocsr()
Donor_te_AvgW2V = hstack((avg_w2v_vectors_test_essays,avg_w2v_vectors_test_title,
Donor_test_state_ohe, Donor_test_teacher_ohe, Donor_test_grade_ohe,
Donor_test_clean_cat_ohe,Donor_test_clean_subcat_ohe,Donor_test_price_norm,Donor_test_postedCount_r
orm,Donor_test_quantity_norm)).tocsr()

print("Final Donor Data Matrix for Set 2")
print(Donor_tr_AvgW2V.shape,Approved_train.shape)
print(Donor_te_AvgW2V.shape,Approved_test.shape)
print("="*100)
```

Final Donor Data Matrix for Set 2  
(73196, 703) (73196,)  
(36052, 703) (36052,)

In [128]:

```
%%time
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

parameters = [{'C': [10**-4, 10**-2, 10**0, 10**2, 10**4]}]

#Using GridSearchCV
modelSet3 = GridSearchCV(LogisticRegression(max_iter=300), parameters, scoring = 'roc_auc', cv=3, r
eturn_train_score = True)
modelSet3.fit(Donor_tr_AvgW2V, Approved_train)

print(modelSet3.best_estimator_)
print(modelSet3.score(Donor_te_AvgW2V, Approved_test))
```

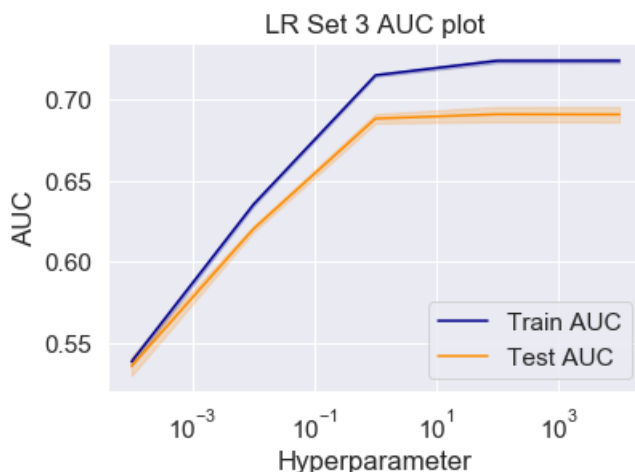
```
LogisticRegression(C=100, class_weight=None, dual=False, fit_intercept=True,
 intercept_scaling=1, max_iter=300, multi_class='warn',
 n_jobs=None, penalty='l2', random_state=None, solver='warn',
 tol=0.0001, verbose=0, warm_start=False)
0.6928250459065574
Wall time: 10min 3s
```

In [129]:

```
Plotting AUC for Train and Test data of Set 3

hyperparams = [10**-4, 10**-2, 10**0, 10**2, 10**4]
set3_train_auc= modelSet3.cv_results_['mean_train_score']
set3_train_auc_std= modelSet3.cv_results_['std_train_score']
set3_test_auc = modelSet3.cv_results_['mean_test_score']
set3_test_auc_std= modelSet3.cv_results_['std_test_score']

plt.figure()
plt.title('LR Set 3 AUC plot')
plt.xlabel('Hyperparameter')
plt.ylabel('AUC')
plt.semilogx(hyperparams, set3_train_auc, label='Train AUC',color='darkblue')
plt.gca().fill_between(hyperparams,set3_train_auc - set3_train_auc_std,set3_train_auc + set3_train_auc_std,alpha=0.2,color='darkblue')
plt.semilogx(hyperparams, set3_test_auc,label='Test AUC', color='darkorange')
plt.gca().fill_between(hyperparams, set3_test_auc - set3_test_auc_std, set3_test_auc + set3_test_auc_std,alpha=0.2,color='darkorange')
plt.legend(loc='best')
plt.show()
```



In [131]:

```
%%time
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

set3Model = LogisticRegression(C=0.01, penalty='l2',max_iter=500);
set3Model.fit(Donor_tr_AvgW2V, Approved_train)

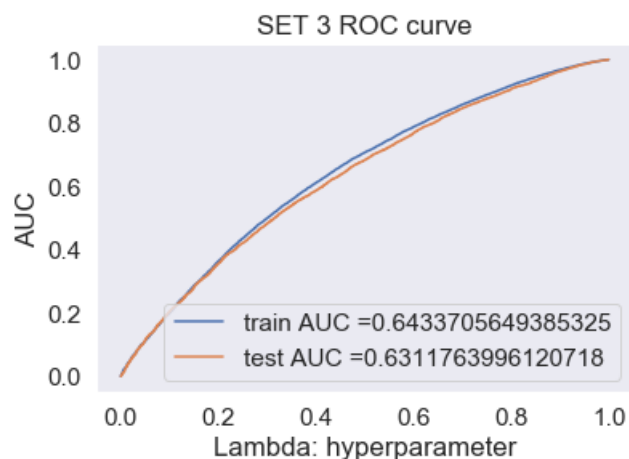
Approved_train_pred_set3 = set3Model.predict_proba(Donor_tr_AvgW2V)
Approved_test_pred_set3 = set3Model.predict_proba(Donor_te_AvgW2V)

train_fpr, train_tpr, tr_thresholds = roc_curve(Approved_train, Approved_train_pred_set3[:, 1])
test_fpr, test_tpr, te_thresholds = roc_curve(Approved_test, Approved_test_pred_set3[:, 1])

plt.plot(train_fpr, train_tpr, label="train AUC =" +str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Lambda: hyperparameter")
plt.ylabel("AUC")
```



```
plt.title("SET 3 ROC curve")
plt.grid()
plt.show()
```



Wall time: 10.9 s

In [132]:

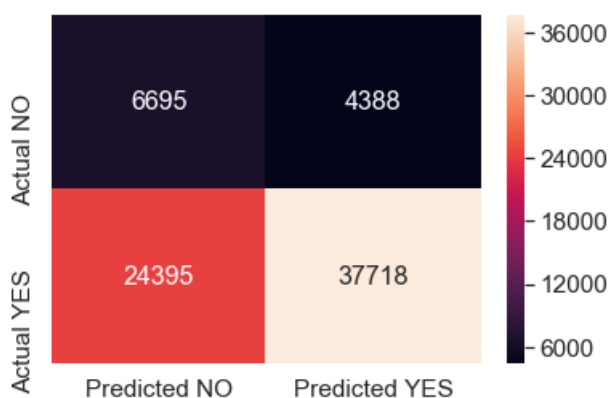
```
Confusion Matrix for Set 3 Train vs Test data

print("Train confusion matrix for Set 3")
donor_SET3_tr = pd.DataFrame(confusion_matrix(Approved_train,
predictcm(Approved_train_pred_set3[:, 1], tr_thresholds, train_fpr, train_tpr)))
donor_SET3_tr.columns = ['Predicted NO', 'Predicted YES']
donor_SET3_tr = donor_SET3_tr.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(donor_SET3_tr, annot=True, annot_kws={"size": 16}, fmt='g')
```

Train confusion matrix for Set 3  
the maximum value of  $tpr \cdot (1 - fpr)$  0.366825398930813 for threshold 0.846

Out[132]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c4376a28d0>



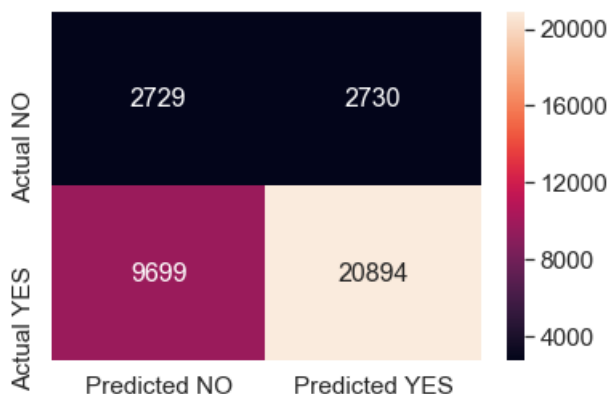
In [133]:

```
print("="*100)
print("Test confusion matrix for Set 3")
donor_SET3_te = pd.DataFrame(confusion_matrix(Approved_test, predict(Approved_test_pred_set3[:,
1], te_thresholds, test_fpr, test_tpr)))
donor_SET3_te.columns = ['Predicted NO', 'Predicted YES']
donor_SET3_te = donor_SET3_te.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(donor_SET3_te, annot=True, annot_kws={"size": 16}, fmt='g')
```

Test confusion matrix for Set 3  
the maximum value of  $tpr \cdot (1 - fpr)$  0.24999999161092998 for threshold 0.837

Out[133]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c437efbc50>



## 2.4.2 Applying Logistic Regression on TFIDF W2V, SET 4

Set 4: categorical, numerical features + project\_title(TFIDF W2V)+ preprocessed\_essay (TFIDF W2V)

In [142]:

```
from scipy.sparse import hstack
Donor_tr_TFIDFW2V = hstack((Donor_train_tfidf_w2v_vectors, Donor_train_tfidf_w2v_title,
Donor_train_state_ohe, Donor_train_teacher_ohe, Donor_train_grade_ohe, Donor_train_clean_cat_ohe,
Donor_train_clean_subcat_ohe, Donor_train_price_norm, Donor_train_postedCount_norm, Donor_train_quantity_norm)).tocsr()
Donor_te_TFIDFW2V = hstack((Donor_test_tfidf_w2v_vectors, Donor_test_tfidf_w2v_title,
Donor_test_state_ohe, Donor_test_teacher_ohe, Donor_test_grade_ohe,
Donor_test_clean_cat_ohe, Donor_test_clean_subcat_ohe, Donor_test_price_norm, Donor_test_postedCount_norm, Donor_test_quantity_norm)).tocsr()

print("Final Donor Data Matrix for Set 2")
print(Donor_tr_TFIDFW2V.shape, Approved_train.shape)
print(Donor_te_TFIDFW2V.shape, Approved_test.shape)
print("="*100)
```

Final Donor Data Matrix for Set 2  
(73196, 703) (73196,)  
(36052, 703) (36052,)

In [143]:

```
%%time
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

parameters = [{'C': [10**-4, 10**-2, 10**0, 10**2, 10**4]}]

#Using GridSearchCV
modelSet4 = GridSearchCV(LogisticRegression(max_iter=300), parameters, scoring = 'roc_auc', cv=3, return_train_score = True)
modelSet4.fit(Donor_tr_TFIDFW2V, Approved_train)

print(modelSet4.best_estimator_)
print(modelSet4.score(Donor_te_TFIDFW2V, Approved_test))
```

LogisticRegression(C=100, class\_weight=None, dual=False, fit\_intercept=True, intercept\_scaling=1, max\_iter=300, multi\_class='warn', n\_jobs=None, penalty='l2', random\_state=None, solver='warn').

```

 n_jobs=None, penalty='l2', random_state=None, solver='saga',
 tol=0.0001, verbose=0, warm_start=False)
0.6815299302059377
Wall time: 22min 59s

```

In [145]:

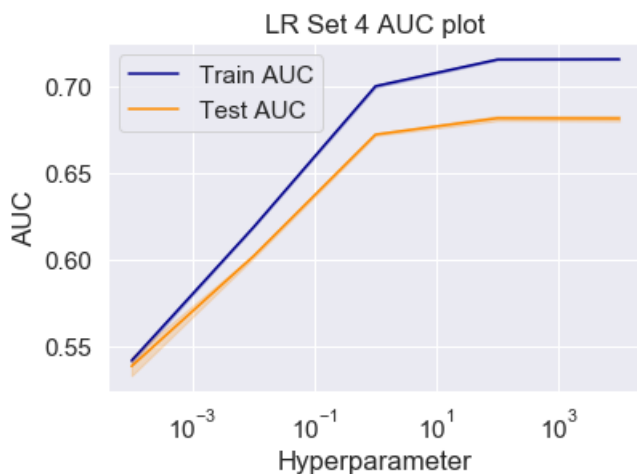
```

Plotting AUC for Train and Test data of Set 4

hyperparams = [10**-4, 10**-2, 10**0, 10**2, 10**4]
set4_train_auc= modelSet4.cv_results_['mean_train_score']
set4_train_auc_std= modelSet4.cv_results_['std_train_score']
set4_test_auc = modelSet4.cv_results_['mean_test_score']
set4_test_auc_std= modelSet4.cv_results_['std_test_score']

plt.figure()
plt.title('LR Set 4 AUC plot')
plt.xlabel('Hyperparameter')
plt.ylabel('AUC')
plt.semilogx(hyperparams, set4_train_auc, label='Train AUC',color='darkblue')
plt.gca().fill_between(hyperparams,set4_train_auc - set4_train_auc_std,set4_train_auc + set4_train_auc_std,alpha=0.2,color='darkblue')
plt.semilogx(hyperparams, set4_test_auc,label='Test AUC', color='darkorange')
plt.gca().fill_between(hyperparams, set4_test_auc - set4_test_auc_std, set4_test_auc + set4_test_auc_std,alpha=0.2,color='darkorange')
plt.legend(loc='best')
plt.show()

```



In [148]:

```

%%time
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

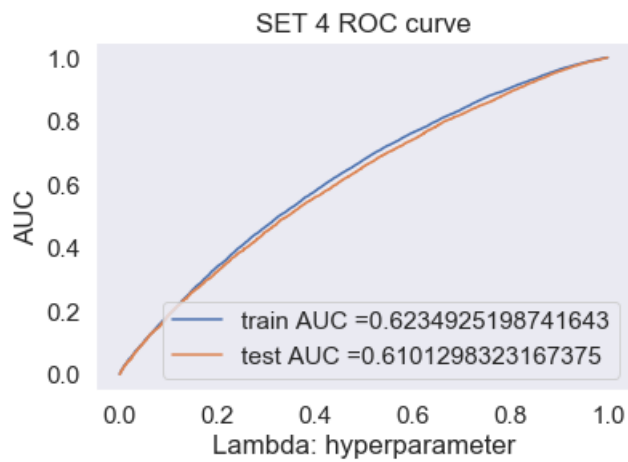
set4Model = LogisticRegression(C=0.01, penalty='l2',max_iter=500);
set4Model.fit(Donor_tr_TFIDFW2V, Approved_train)

Approved_train_pred_set4 = set4Model.predict_proba(Donor_tr_TFIDFW2V)
Approved_test_pred_set4 = set4Model.predict_proba(Donor_te_TFIDFW2V)

train_fpr, train_tpr, tr_thresholds = roc_curve(Approved_train, Approved_train_pred_set4[:, 1])
test_fpr, test_tpr, te_thresholds = roc_curve(Approved_test, Approved_test_pred_set4[:, 1])

plt.plot(train_fpr, train_tpr, label="train AUC =" +str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Lambda: hyperparameter")
plt.ylabel("AUC")
plt.title("SET 4 ROC curve")
plt.grid()
plt.show()

```



Wall time: 17.5 s

In [149]:

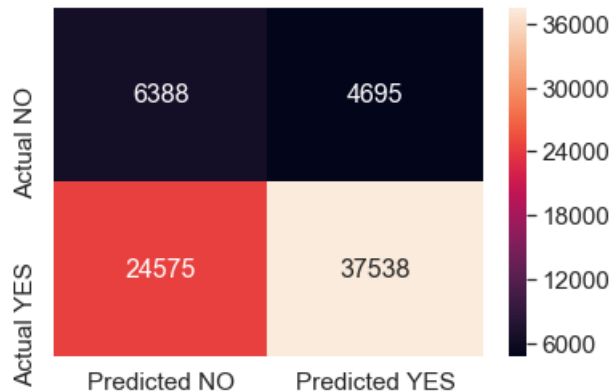
```
Confusion Matrix for Set 5 Train vs Test data
```

```
print("Train confusion matrix for Set 4")
donor_SET4_tr = pd.DataFrame(confusion_matrix(Approved_train,
predictcm(Approved_train_pred_set4[:, 1], tr_thresholds, train_fpr, train_tpr)))
donor_SET4_tr.columns = ['Predicted NO', 'Predicted YES']
donor_SET4_tr = donor_SET4_tr.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(donor_SET4_tr, annot=True, annot_kws={"size": 16}, fmt='g')
```

Train confusion matrix for Set 4  
the maximum value of  $tpr \cdot (1 - fpr)$  0.3483342659062246 for threshold 0.846

Out[149]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c400464a58>



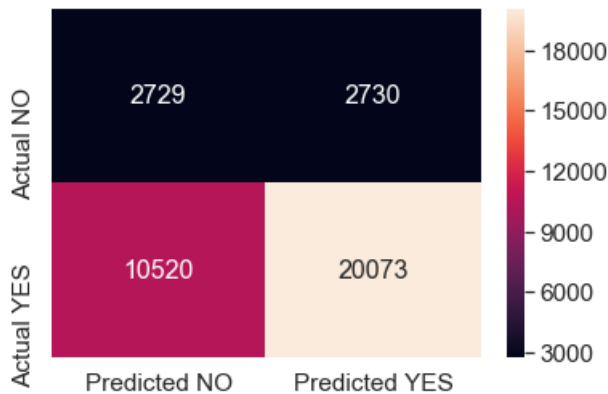
In [150]:

```
print("="*100)
print("Test confusion matrix for Set 4")
donor_SET4_te = pd.DataFrame(confusion_matrix(Approved_test, predict(Approved_test_pred_set4[:,
1], te_thresholds, test_fpr, test_fpr)))
donor_SET4_te.columns = ['Predicted NO', 'Predicted YES']
donor_SET4_te = donor_SET4_te.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(donor_SET4_te, annot=True, annot_kws={"size": 16}, fmt='g')
```

Test confusion matrix for Set 4  
the maximum value of  $tpr \cdot (1 - fpr)$  0.24999999161092998 for threshold 0.84

Out[150]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c4394af2b0>



## 2.5 Logistic Regression with added Features `Set 5`

In [105]:

```
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
Donor_tr_set5 = hstack((Donor_train_state_ohe, Donor_train_teacher_ohe, Donor_train_grade_ohe,
Donor_train_clean_cat_ohe, Donor_train_clean_subcat_ohe, Donor_train_price_norm, Donor_train_postedCount_norm, Donor_train_quantity_norm)).tocsr()
Donor_te_set5 = hstack((Donor_test_state_ohe, Donor_test_teacher_ohe, Donor_test_grade_ohe,
Donor_test_clean_cat_ohe, Donor_test_clean_subcat_ohe, Donor_test_price_norm, Donor_test_postedCount_norm, Donor_test_quantity_norm)).tocsr()

print("Final Donor Data Matrix for Set 1")
print(Donor_tr_set5.shape, Approved_train.shape)
print(Donor_te_set5.shape, Approved_test.shape)
print("="*100)
```

```
Final Donor Data Matrix for Set 1
(73196, 103) (73196,)
(36052, 103) (36052,)
```

In [109]:

```
%%time
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

parameters = [{'C': [10**-4, 10**-2, 10**0, 10**2, 10**4]}]

#Using GridSearchCV
modelSet5 = GridSearchCV(LogisticRegression(max_iter=300), parameters, scoring = 'roc_auc', cv=3, return_train_score = True)
modelSet5.fit(Donor_tr_set5, Approved_train)

print(modelSet5.best_estimator_)
print(modelSet5.score(Donor_te_set5, Approved_test))
```

```
LogisticRegression(C=0.01, class_weight=None, dual=False, fit_intercept=True,
 intercept_scaling=1, max_iter=300, multi_class=warn,
```

```

intercept_scaling=1, max_iter=500, multi_class='warn',
n_jobs=None, penalty='l2', random_state=None, solver='warn',
tol=0.0001, verbose=0, warm_start=False)
0.5712479667117559
Wall time: 10.2 s

```

In [110]:

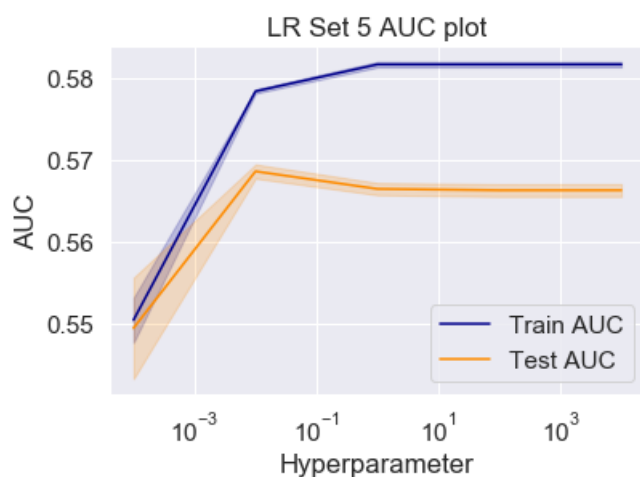
```

Plotting AUC for Train and Test data of Set 5

hyperparams = [10**-4, 10**-2, 10**0, 10**2, 10**4]
set5_train_auc= modelSet5.cv_results_['mean_train_score']
set5_train_auc_std= modelSet5.cv_results_['std_train_score']
set5_test_auc = modelSet5.cv_results_['mean_test_score']
set5_test_auc_std= modelSet5.cv_results_['std_test_score']

plt.figure()
plt.title('LR Set 5 AUC plot')
plt.xlabel('Hyperparameter')
plt.ylabel('AUC')
plt.semilogx(hyperparams, set5_train_auc, label='Train AUC',color='darkblue')
plt.gca().fill_between(hyperparams,set5_train_auc - set5_train_auc_std,set5_train_auc + set5_train_auc_std,alpha=0.2,color='darkblue')
plt.semilogx(hyperparams, set5_test_auc,label='Test AUC', color='darkorange')
plt.gca().fill_between(hyperparams, set5_test_auc - set5_test_auc_std, set5_test_auc + set5_test_auc_std,alpha=0.2,color='darkorange')
plt.legend(loc='best')
plt.show()

```



In [114]:

```

%%time
https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import *
from sklearn.linear_model import LogisticRegression

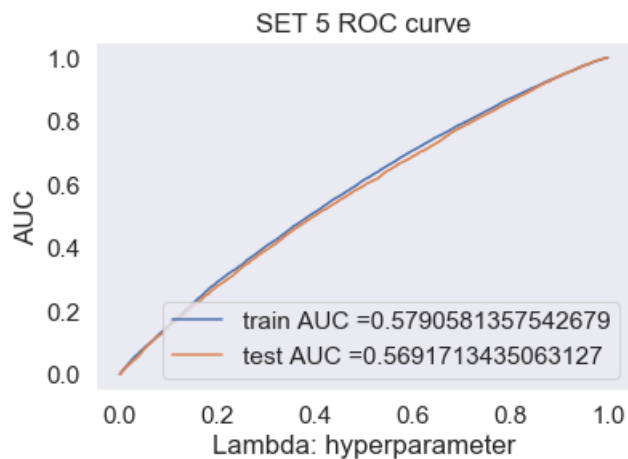
set5Model = LogisticRegression(C=100, penalty='l2',max_iter=500);
set5Model.fit(Donor_tr_set5, Approved_train)

Approved_train_pred_set5 = set5Model.predict_proba(Donor_tr_set5)
Approved_test_pred_set5 = set5Model.predict_proba(Donor_te_set5)

train_fpr, train_tpr, tr_thresholds = roc_curve(Approved_train, Approved_train_pred_set5[:, 1])
test_fpr, test_tpr, te_thresholds = roc_curve(Approved_test, Approved_test_pred_set5[:, 1])

plt.plot(train_fpr, train_tpr, label="train AUC =" +str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" +str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("Lambda: hyperparameter")
plt.ylabel("AUC")
plt.title("SET 5 ROC curve")
plt.grid()
plt.show()

```



Wall time: 1.99 s

In [115]:

```
Confusion Matrix for Set 5 Train vs Test data
```

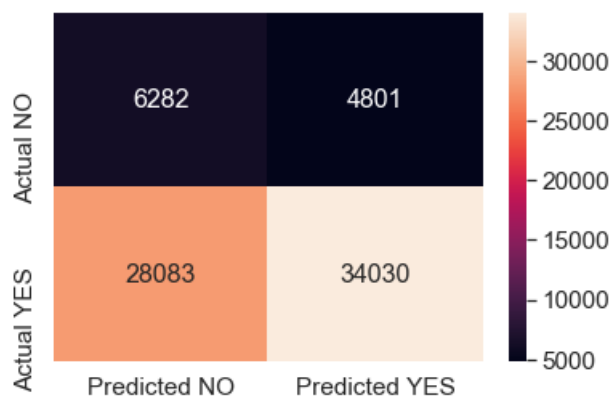
```
print("Train confusion matrix for Set 2")
donor_SET5_tr = pd.DataFrame(confusion_matrix(Approved_train,
predictcm(Approved_train_pred_set5[:, 1], tr_thresholds, train_fpr, train_tpr)))
donor_SET5_tr.columns = ['Predicted NO', 'Predicted YES']
donor_SET5_tr = donor_SET5_tr.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(donor_SET5_tr, annot=True, annot_kws={"size": 16}, fmt='g')
```

Train confusion matrix for Set 2

the maximum value of  $tpr \cdot (1 - fpr)$  0.3105417829579172 for threshold 0.85

Out[115]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c400132ef0>



In [117]:

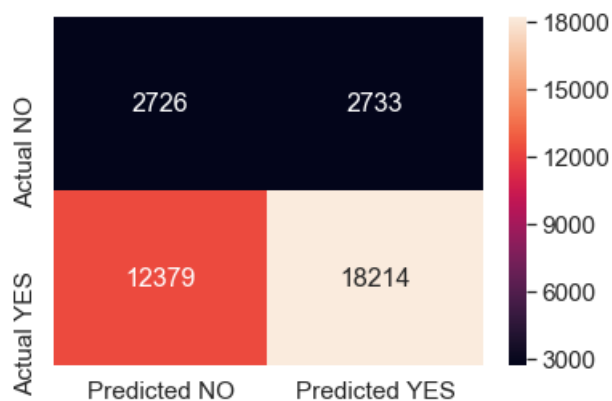
```
print("="*100)
print("Test confusion matrix for Set 5")
donor_SET5_te = pd.DataFrame(confusion_matrix(Approved_test, predict(Approved_test_pred_set5[:,
1], te_thresholds, test_fpr, test_fpr)))
donor_SET5_te.columns = ['Predicted NO', 'Predicted YES']
donor_SET5_te = donor_SET5_te.rename({0: 'Actual NO', 1: 'Actual YES'})
sns.set(font_scale=1.4)
sns.heatmap(donor_SET5_te, annot=True, annot_kws={"size": 16}, fmt='g')
```

Test confusion matrix for Set 5

the maximum value of  $tpr \cdot (1 - fpr)$  0.24999958893556828 for threshold 0.846

Out[117]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1c402c03860>



### 3. Conclusion

In [152]:

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

```
from prettytable import PrettyTable
```

```
x = PrettyTable()
```

```
x.field_names = ["Vectorizer", "Model", "Penalty", "Hyper Parameter", "Train AUC", "Test AUC"]
```

```
x.add_row(["Set 1", "Logistic Regression", "L2", 100, 84.9, 67.4])
```

```
x.add_row(["Set 2", "Logistic Regression", "L2", 1, 82.4, 71.1])
```

```
x.add_row(["Set 3", "Logistic Regression", "L2", 0.01, 64.3, 63.1])
```

```
x.add_row(["Set 4", "Logistic Regression", "L2", 0.01, 62.3, 61.0])
```

```
x.add_row(["Set 5", "Logistic Regression", "L2", 100, 57.9, 56.9])
```

```
print(x)
```

| Vectorizer | Model               | Penalty | Hyper Parameter | Train AUC | Test AUC |
|------------|---------------------|---------|-----------------|-----------|----------|
| Set 1      | Logistic Regression | L2      | 100             | 84.9      | 67.4     |
| Set 2      | Logistic Regression | L2      | 1               | 82.4      | 71.1     |
| Set 3      | Logistic Regression | L2      | 0.01            | 64.3      | 63.1     |
| Set 4      | Logistic Regression | L2      | 0.01            | 62.3      | 61.0     |
| Set 5      | Logistic Regression | L2      | 100             | 57.9      | 56.9     |