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	I
`project_id`	A unique identifier for the proposed project. **Example:**
`project_title`	Title of the project. • `Art Will Make Yc` • `First G`
`project_grade_category`	Grade level of students for which the project is targeted. One of t enumera • `Grade` • `Gr` • `Gr` • `Gra`

Feature	I
project_subject_categories	One or more (comma-separated) subject categories for the proj following enumerated lis • `Applied` • `Care` • `Health` • `History` • `Literacy &` • `Math &` • `Music &` • `Speci` • • `Music &` • `Literacy & Language, Math &`
school_state	State where school is located ([Two-letter U.S. (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos) **Ex**
project_subject_subcategories	One or more (comma-separated) subject subcategories for • • `Literature & Writing, Social`
project_resource_summary	An explanation of the resources needed for the project • `My students need hands on literacy materials sensory need`
project_essay_1	First applic
project_essay_2	Second applic
project_essay_3	Third applic
project_essay_4	Fourth applic
project_submitted_datetime	Datetime when project application was submitted. **Example:** `20 12:4`
teacher_id	A unique identifier for the teacher of the proposed projec `bdf8baa8fedef6bfeec7ae4f`
teacher_prefix	Teacher's title. One of the following enumera • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the sa **E**

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

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

Feature	Description
id	A `project_id` value from the `train.csv` file. **Example:** `p036502`
description	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`

Feature	Description
`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
from scipy.sparse import hstack
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn import preprocessing
from sklearn.metrics import confusion_matrix
from prettytable import PrettyTable
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from sklearn.ensemble import RandomForestClassifier
from xgboost.sklearn import XGBClassifier
```

1.1 Reading Data

In [2]:

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

Adding price attribute to project_data dataframe from resources using merge function

In [3]:

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

In [4]:

```
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, 19)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_pref
ix' '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'
 'price' 'quantity']
```

In [5]:

```
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[5]:

	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 [6]:

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-
cat_list = []
for i in catogories:
    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", "Wa
        if 'The' in j.split(): # this will split each of the catogory based on spac
            j=j.replace('The','') # if we have the words "The" we are going to repl
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) e
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [7]:

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

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

sub_cat_list = []
for i in sub_catogories:
    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", "Wa
        if 'The' in j.split(): # this will split each of the catogory based on spac
            j=j.replace('The','') # if we have the words "The" we are going to repl
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) e
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing
        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/22898595/408
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 [8]:

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

In [9]:

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

Out[9]:

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

In [10]:

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

In [11]:

```
# 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, immigrant
s, and native-born Americans bringing the gift of language to our scho
ol. \r\n\r\n We have over 24 languages represented in our English Lear
ner program with students at every level of mastery.  We also have ove
r 40 countries represented with the families within our school.  Each
student brings a wealth of knowledge and experiences to us that open o
ur eyes to new cultures, beliefs, and respect.\"The limits of your lan
guage are the limits of your world.\"-Ludwig Wittgenstein  Our English
learner's have a strong support system at home that begs for more reso
urces.  Many times our parents are learning to read and speak English
along side of their children.  Sometimes this creates barriers for par
ents to be able to help their child learn phonetics, letter recognitio
n, and other reading skills.\r\n\r\nBy providing these dvd's and playe
rs, students are able to continue their mastery of the English languag
e even if no one at home is able to assist.  All families with student
s within the Level 1 proficiency status, will be a offered to be a par
t of this program.  These educational videos will be specially chosen
by the English Learner Teacher and will be sent home regularly to watc
h.  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 oppo
rtunity 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\nnannan
==================================================
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 stu
dents, 97.3% are minority students. \r\nThe school has a vibrant commu
nity that loves to get together and celebrate. Around Halloween there
is a whole school parade to show off the beautiful costumes that stude
nts wear. On Cinco de Mayo we put on a big festival with crafts made b
y the students, dances, and games. At the end of the year the school h
osts a carnival to celebrate the hard work put in during the school ye
ar, with a dunk tank being the most popular activity.My students will
use these five brightly colored Hokki stools in place of regular, stat
ionary, 4-legged chairs. As I will only have a total of ten in the cla
ssroom and not enough for each student to have an individual one, they
will be used in a variety of ways. During independent reading time the
y 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 m
ath and reading times. The rest of the day they will be used by the st
udents 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 thei
r 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 t
```

```
he 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 al
ways students who head over to the kidney table to get one of the stoo
ls who are disappointed as there are not enough of them. \r\n\r\nWe as
k 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 sa
me time. These stools will help students to meet their 60 minutes a da
y of movement by allowing them to activate their core muscles for bala
nce while they sit. For many of my students, these chairs will take aw
ay 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 environme
nt with plain walls, rows of desks, and a teacher in front of the roo
m? A typical day in our room is nothing like that. I work hard to crea
te 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-pric
e 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 spon
ges, 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 abl
e to help create the mood in our classroom setting to be one of a them
ed nautical environment. Creating a classroom environment is very impo
rtant 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 cl
assroom for the first time on Meet the Teacher evening. I'll take pict
ures of each child with them, have them developed, and then hung in ou
r 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 th
ank 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 dona
tions will help me to help make our classroom a fun, inviting, learnin
g environment from day one.\r\n\r\nIt costs lost of money out of my ow
n pocket on resources to get our classroom ready. Please consider help
ing with this project to make our new school year a very successful on
e. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech
and language delays, cognitive delays, gross/fine motor delays, to aut
ism. They are eager beavers and always strive to work their hardest wo
rking past their limitations. \r\n\r\nThe materials we have are the on
es 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 di
sabilities and limitations, my students love coming to school and come
eager to learn and explore.Have you ever felt like you had ants in you
r pants and you needed to groove and move as you were in a meeting? Th
is is how my kids feel all the time. The want to be able to move as th
ey learn or so they say.Wobble chairs are the answer and I love then b
ecause 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 ju
mping 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 des
erves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior te
```

```
acher demonstrates. The great teacher inspires. -William A. Ward\r\n\r
\nMy school has 803 students which is makeup is 97.6% African-America
n, 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 s
tudents are on free or reduced lunch. We aren't receiving doctors, law
yers, or engineers children from rich backgrounds or neighborhoods. As
an educator I am inspiring minds of young children and we focus not on
ly on academics but one smart, effective, efficient, and disciplined s
tudents with good character.In our classroom we can utilize the Blueto
oth 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 st
udents 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 thing
s 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 stud
ents to learn about different letters and it is more accessible.nannan
==================================================
```

In [12]:

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

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

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

In [13]:

```
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 aut
ism. They are eager beavers and always strive to work their hardest wo
rking past their limitations. \r\n\r\nThe materials we have are the on
es 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 di
sabilities and limitations, my students love coming to school and come
eager to learn and explore.Have you ever felt like you had ants in you
r pants and you needed to groove and move as you were in a meeting? Th
is is how my kids feel all the time. The want to be able to move as th
ey learn or so they say.Wobble chairs are the answer and I love then b
ecause 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 j
umping 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 des
erves.nannan
==================================================
```

In [14]:

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

```
My kindergarten students have varied disabilities ranging from speech
and language delays, cognitive delays, gross/fine motor delays, to aut
ism. They are eager beavers and always strive to work their hardest wo
rking 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 th
e students receive free or reduced price lunch.  Despite their disabil
ities and limitations, my students love coming to school and come eage
r to learn and explore.Have you ever felt like you had ants in your pa
nts and you needed to groove and move as you were in a meeting? This i
s how my kids feel all the time. The want to be able to move as they l
earn or so they say.Wobble chairs are the answer and I love then becau
se they develop their core, which enhances gross motor and in Turn fin
e motor skills.   They also want to learn through games, my kids do no
t want to sit and do worksheets. They want to learn to count by jumpin
g and playing. Physical engagement is the key to our success. The numb
er toss and color and shape mats can make that happen. My students wil
l forget they are doing work and just have the fun a 6 year old deserv
es.nannan
```

In [15]:

```
#remove spacial 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 f
or my students I teach in a Title I school where most of the students
receive free or reduced price lunch Despite their disabilities and lim
itations 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 nee
ded to groove and move as you were in a meeting This is how my kids fe
el all the time The want to be able to move as they learn or so they s
ay Wobble chairs are the answer and I love then because they develop t
heir core which enhances gross motor and in Turn fine motor skills The
y also want to learn through games my kids do not want to sit and do w
orksheets They want to learn to count by jumping and playing Physical
engagement is the key to our success The number toss and color and sha
pe mats can make that happen My students will forget they are doing wo
rk and just have the fun a 6 year old deserves nannan
```

In [16]:

```
# 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', "yo
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'hi
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that',
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because',
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'thro
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', '
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've",
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', '
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't"
            'won', "won't", 'wouldn', "wouldn't"}
```

In [17]:

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

```
100%|██████████| 109248/109248 [00:14<00:00, 7296.50it/s]
```

In [18]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[18]:

```
'kindergarten students varied disabilities ranging speech language del
ays cognitive delays gross fine motor delays autism eager beavers alwa
ys strive work hardest working past limitations the materials ones see
k students teach title school students receive free reduced price lunc
h despite disabilities limitations students love coming school come ea
ger learn explore have ever felt like ants pants needed groove move me
eting kids feel time want able move learn say wobble chairs answer lov
e develop core enhances gross motor turn fine motor skills they also w
ant learn games kids want sit worksheets want learn count jumping play
ing physical engagement key success number toss color shape mats make
happen students forget work fun 6 year old deserves nannan'
```

In [19]:

```
project_data['clean_essay'] = preprocessed_essays
```

In [20]:

```
project_data.drop(['project_essay_1','project_essay_2','project_essay_3','project_e
```

1.4 Preprocessing of `project_title`

- Decontract project titles, remove line breaks and extra spaces, convert everything to lowercase and then remove all the stop words.

In [21]:

```
preprocessed_titles = []

for title in tqdm(project_data['project_title'].values):
    title = title.lower().strip()
    title = ' '.join(e for e in title.split() if e.lower() not in stopwords)
    title = decontracted(title)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    preprocessed_titles.append(title)
```

```
100%|██████████| 109248/109248 [00:01<00:00, 67698.27it/s]
```

In [22]:

```
project_data['clean_title'] = preprocessed_titles
project_data.drop(['project_title'],axis=1,inplace=True)
```

Pre-processing teacher_prefix

In [23]:

```
#remove nan from teacher prefix:
#https://stackoverflow.com/questions/21011777/how-can-i-remove-nan-from-list-python
def remove_nan(prefix):
    if str(prefix)!='nan':
        pr = str(prefix)
        pr = re.sub("\\.","",pr) #remove dot from the end of prefix
        return pr
    return "none"

cleaned_teacher_prefix = project_data['teacher_prefix'].map(remove_nan)
project_data['clean_teacher_prefix'] = cleaned_teacher_prefix
```

In [24]:

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

Pre-process project_grade_category

- Clean the project grade categories:
    - Convert `Grades 3-5 ==> Grades_3_5`

In [25]:

```
def clean_project_grades(grade):
    grade = re.sub("\-","_",grade)
    grade = re.sub(" ","_",grade)
    return grade.strip()

clean_grades = project_data['project_grade_category'].map(clean_project_grades)
project_data['clean_grade_category'] = clean_grades
```

In [26]:

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

In [27]:

```
# Dropping all features we won't need going forward
project_data.drop(['project_resource_summary'],axis=1,inplace=True)
project_data.drop(['Unnamed: 0','teacher_id'],axis=1,inplace=True)
```

In [28]:

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

Out[28]:

	id	school_state	project_submitted_datetime	teacher_number_of_previously_posted_proj
0	p253737	IN	2016-12-05 13:43:57	
1	p258326	FL	2016-10-25 09:22:10	

Assignment 9: RF and GBDT

Response Coding: Example

> The response tabel is built only on train dataset. For a category which is not there in train data and present in test data, we will encode them with default values Ex: in our test data if have State: D then we encode it as [0.5, 0.05]

1. **Apply both Random Forrest and GBDT on these feature sets**

- Set 1: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(BOW) + preprocessed_eassay (BOW)
- Set 2: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
- Set 3: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
- Set 4: categorical(instead of one hot encoding, try response coding (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/): use probability values), numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

2. **The hyper paramter tuning (Consider any two hyper parameters preferably n_estimators, max_depth)**

- Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
- find the best hyper paramter using k-fold cross validation/simple cross validation data
- use gridsearch cv or randomsearch cv or you can write your own for loops to do this task

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

  with X-axis as **n_estimators**, Y-axis as **max_depth**, and Z-axis as **AUC Score** , we have given the notebook which explains how to plot this 3d plot, you can find it in the same drive *3d_scatter_plot.ipynb*

or

- 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

  seaborn heat maps (https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**
- You can choose either of the plotting techniques: 3d plot or heat map
- 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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points

4. **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 (http://zetcode.com/python/prettytable/)

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. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2. Random Forest and GBDT

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

In [29]:

```
#Separating features and label column
Y = project_data['project_is_approved']
X = project_data.drop(['project_is_approved','id'],axis=1)
print("Shape of X: ",X.shape)
print("Shape of Y: ",Y.shape)
```

```
Shape of X:  (109248, 12)
Shape of Y:  (109248,)
```

In [30]:

```
#separating data into train and test
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.30,stratify=Y)
print("Shape of X_train: ", X_train.shape)
print("Shape of Y_train: ",Y_train.shape)
print("Shape of X_test: ",X_test.shape)
print("Shape of Y_test: ",Y_test.shape)
```

```
Shape of X_train:  (76473, 12)
Shape of Y_train:  (76473,)
Shape of X_test:  (32775, 12)
Shape of Y_test:  (32775,)
```

In [31]:

```
X_train.columns
```

Out[31]:

```
Index(['school_state', 'project_submitted_datetime',
       'teacher_number_of_previously_posted_projects', 'price', 'quant
ity',
       'clean_categories', 'clean_subcategories', 'essay', 'clean_essa
y',
       'clean_title', 'clean_teacher_prefix', 'clean_grade_category'],
      dtype='object')
```

2.2 Make Data Model Ready: encoding numerical, categorical features

2.2.1 Encoding Categorical Features

Response Coding

In [32]:

```
class ResponseCoding:

    def __init__(self):
        self.proba = {}

    def fit(self,feature,label):
        '''
            Takes in a categorical feature and learns the response coding by buildi
            Each category is mapped to a probability p = P(C=c|label=1)
        '''
        df = pd.DataFrame()
        df['feature'] = feature
        df['label'] = label

        unique_categories = np.unique(feature)

        for category in unique_categories:
            n_values = df[df['feature']==category].count()['feature']
            cat_count_0 = df[(df['feature']==category)& (df['label']==0)].count()['
            cat_count_1 = df[(df['feature']==category)& (df['label']==1)].count()['
            self.proba[category] = (cat_count_0/n_values,cat_count_1/n_values)

    def transform(self,feature):
        '''
            Return the feature transformation based on the probablity values calcul
            If a category/level does not exist, 0.5 is returned.
        '''
        transformed_feature = []

        for category in feature:
            temp = []
            if category in self.proba.keys():
                temp.append(self.proba[category][0])
                temp.append(self.proba[category][1])
                transformed_feature.append(temp)
            else:
                transformed_feature.append([0.5,0.5])
        return transformed_feature

    def get_feature_response_coding(self):
        return self.proba
```

Unit Testing

In [33]:

```
vec = ResponseCoding()
f = ['mr','mr', 'mrs','mrs','dr','dr','dr','dr']
l = [0,1,0,0,1,1,1,0]
vec.fit(f,l)
vec.proba
```

Out[33]:

```
{'dr': (0.25, 0.75), 'mr': (0.5, 0.5), 'mrs': (1.0, 0.0)}
```

In [34]:

```
res = vec.transform(['mr','mrs','dr'])
res
```

Out[34]:

```
[[0.5, 0.5], [1.0, 0.0], [0.25, 0.75]]
```

In [35]:

```
arr  = np.array(res)
arr[1][0], arr[1][1]
```

Out[35]:

```
(1.0, 0.0)
```

Response Coding: clean_categories

In [36]:

```
vectorizer_category = ResponseCoding()
vectorizer_category.fit(X_train['clean_categories'].values,Y_train.values)

X_train_category_response = np.array(vectorizer_category.transform(X_train['clean_c
X_test_category_response = np.array(vectorizer_category.transform(X_test['clean_cat

print(vectorizer_category.get_feature_response_coding())
```

```
{'AppliedLearning': (0.183206106870229, 0.816793893129771), 'AppliedLe
arning Health_Sports': (0.17535545023696683, 0.8246445497630331), 'App
liedLearning History_Civics': (0.18253968253968253, 0.817460317460317
4), 'AppliedLearning Literacy_Language': (0.1395498392282958, 0.860450
1607717042), 'AppliedLearning Math_Science': (0.18488529014844804, 0.8
15114709851552), 'AppliedLearning Music_Arts': (0.20326678765880218,
0.7967332123411979), 'AppliedLearning SpecialNeeds': (0.18950437317784
258, 0.8104956268221575), 'AppliedLearning Warmth Care_Hunger': (0.142
85714285714285, 0.8571428571428571), 'Health_Sports': (0.1450892857142
8573, 0.8549107142857143), 'Health_Sports AppliedLearning': (0.1653543
3070866143, 0.8346456692913385), 'Health_Sports History_Civics': (0.1,
0.9), 'Health_Sports Literacy_Language': (0.15384615384615385, 0.84615
38461538461), 'Health_Sports Math_Science': (0.1711229946524064, 0.828
8770053475936), 'Health_Sports Music_Arts': (0.1724137931034483, 0.827
5862068965517), 'Health_Sports SpecialNeeds': (0.12873326467559218, 0.
8712667353244078), 'Health_Sports Warmth Care_Hunger': (0.05, 0.95),
'History_Civics': (0.16842105263157894, 0.8315789473684211), 'History_
Civics AppliedLearning': (0.23333333333333334, 0.7666666666666667), 'H
istory_Civics Health_Sports': (0.14285714285714285, 0.857142857142857
1), 'History_Civics Literacy_Language': (0.09410548086866598, 0.905894
519131334), 'History_Civics Math_Science': (0.1574468085106383, 0.8425
531914893617), 'History_Civics Music_Arts': (0.18340611353711792, 0.81
65938864628821), 'History_Civics SpecialNeeds': (0.18497109826589594,
0.815028901734104), 'Literacy_Language': (0.13140908537327975, 0.86859
09146267202), 'Literacy_Language AppliedLearning': (0.15, 0.85), 'Lite
racy_Language Health_Sports': (0.2619047619047619, 0.738095238095238
1), 'Literacy_Language History_Civics': (0.13345195729537365, 0.866548
0427046264), 'Literacy_Language Math_Science': (0.1335549472037544, 0.
8664450527962456), 'Literacy_Language Music_Arts': (0.175145954962468
4, 0.8248540450375312), 'Literacy_Language SpecialNeeds': (0.143578643
57864358, 0.8564213564213564), 'Literacy_Language Warmth Care_Hunger':
(0.25, 0.75), 'Math_Science': (0.18153743427092897, 0.818462565729071
1), 'Math_Science AppliedLearning': (0.1566125290023201, 0.8433874709
976799), 'Math_Science Health_Sports': (0.22, 0.78), 'Math_Science His
tory_Civics': (0.14583333333333334, 0.8541666666666666), 'Math_Science
Literacy_Language': (0.139549436795995, 0.860450563204005), 'Math_Scie
nce Music_Arts': (0.1607773851590106, 0.8392226148409894), 'Math_Scien
ce SpecialNeeds': (0.17815646785437644, 0.8218435321456236), 'Math_Sci
ence Warmth Care_Hunger': (0.5, 0.5), 'Music_Arts': (0.141464773043720
4, 0.8585352269562796), 'Music_Arts AppliedLearning': (0.4, 0.6), 'Mus
ic_Arts Health_Sports': (0.46153846153846156, 0.5384615384615384), 'Mu
sic_Arts History_Civics': (0.4166666666666667, 0.5833333333333334), 'M
usic_Arts SpecialNeeds': (0.12121212121212122, 0.8787878787878788), 'M
usic_Arts Warmth Care_Hunger': (1.0, 0.0), 'SpecialNeeds': (0.18322368
421052632, 0.8167763157894737), 'SpecialNeeds Health_Sports': (0.21428
571428571427, 0.7857142857142857), 'SpecialNeeds Music_Arts': (0.16587
677725118483, 0.8341232227488151), 'SpecialNeeds Warmth Care_Hunger':
(0.1875, 0.8125), 'Warmth Care_Hunger': (0.07709011943539631, 0.922909
8805646037)}
```

Response Coding: clean_subcategories

In [37]:

```
vectorizer_subcategory = ResponseCoding()
vectorizer_subcategory.fit(X_train['clean_subcategories'].values,Y_train.values)

X_train_subcategory_response = np.array(vectorizer_subcategory.transform(X_train['c
X_test_subcategory_response = np.array(vectorizer_subcategory.transform(X_test['cle

print(vectorizer_subcategory.get_feature_response_coding())
```

```
{'AppliedSciences': (0.18575498575498575, 0.8142450142450143), 'Appl
iedSciences CharacterEducation': (0.11428571428571428, 0.88571428571
42857), 'AppliedSciences Civics_Government': (0.25, 0.75), 'AppliedS
ciences College_CareerPrep': (0.20567375886524822, 0.794326241134751
8), 'AppliedSciences CommunityService': (0.06666666666666667, 0.9333
333333333333), 'AppliedSciences ESL': (0.12727272727272726, 0.872727
2727272727), 'AppliedSciences EarlyDevelopment': (0.1680672268907563
2, 0.8319327731092437), 'AppliedSciences Economics': (0.333333333333
3333, 0.6666666666666666), 'AppliedSciences EnvironmentalScience':
(0.20373027259684362, 0.7962697274031564), 'AppliedSciences Extracur
ricular': (0.08823529411764706, 0.9117647058823529), 'AppliedScience
s FinancialLiteracy': (0.0, 1.0), 'AppliedSciences ForeignLanguage
s': (0.5, 0.5), 'AppliedSciences Gym_Fitness': (0.08333333333333333,
0.9166666666666666), 'AppliedSciences Health_LifeScience': (0.197368
42105263158, 0.8026315789473685), 'AppliedSciences Health_Wellness':
(0.16666666666666666, 0.8333333333333334), 'AppliedSciences History_
Geography': (0.21739130434782608, 0.782608695652174), 'AppliedScienc
es Literacy': (0.17506631299734748, 0.8249336870026526), 'AppliedSci
ences Literature_Writing': (0.13448275862068965, 0.865517241379310
3), 'AppliedSciences Mathematics': (0.16505263157894737, 0.834947368
```

Response Coding: school_state

In [38]:

```
vectorizer_state = ResponseCoding()
vectorizer_state.fit(X_train['school_state'].values,Y_train.values)

X_train_school_state_response = np.array(vectorizer_state.transform(X_train['school
X_test_school_state_response = np.array(vectorizer_state.transform(X_test['school_s

print(vectorizer_state.get_feature_response_coding())
```

```
{'AK': (0.16666666666666666, 0.8333333333333334), 'AL': (0.14946325350
94963, 0.8505367464905037), 'AR': (0.1675603217158177, 0.8324396782841
823), 'AZ': (0.16344229486324216, 0.8365577051367579), 'CA': (0.139274
23059255856, 0.8607257694074414), 'CO': (0.1511627906976744, 0.8488372
093023255), 'CT': (0.12822719449225473, 0.8717728055077453), 'DC': (0.
1925133689839572, 0.8074866310160428), 'DE': (0.100418410041841, 0.899
581589958159), 'FL': (0.1666278256816593, 0.8333721743183408), 'GA':
(0.16041443372633082, 0.8395855662736692), 'HI': (0.1510574018126888,
0.8489425981873112), 'IA': (0.13859275053304904, 0.8614072494669509),
'ID': (0.15756302521008403, 0.842436974789916), 'IL': (0.1499503475670
308, 0.8500496524329693), 'IN': (0.15359477124183007, 0.84640522875816
99), 'KS': (0.16930022573363432, 0.8306997742663657), 'KY': (0.1388888
888888889, 0.8611111111111112), 'LA': (0.17300832342449465, 0.82699167
65755053), 'MA': (0.13560334528076465, 0.8643966547192353), 'MD': (0.1
560418648905804, 0.8439581351094196), 'ME': (0.14912280701754385, 0.85
08771929824561), 'MI': (0.15468607825295724, 0.8453139217470428), 'M
N': (0.13784764207980654, 0.8621523579201935), 'MO': (0.14079020589872
01, 0.85920979410128), 'MS': (0.15829694323144106, 0.841703056768558
9), 'MT': (0.18128654970760233, 0.8187134502923976), 'NC': (0.14281713
80565668, 0.8571828619434332), 'ND': (0.10679611650485436, 0.893203883
4951457), 'NE': (0.17040358744394618, 0.8295964125560538), 'NH': (0.13
671875, 0.86328125), 'NJ': (0.15832805573147563, 0.8416719442685244),
'NM': (0.14690721649484537, 0.8530927835051546), 'NV': (0.143776824034
33475, 0.8562231759656652), 'NY': (0.13903432228039558, 0.860965677719
6044), 'OH': (0.1339031339031339, 0.8660968660968661), 'OK': (0.169704
58830923948, 0.8302954116907605), 'OR': (0.15977653631284916, 0.840223
4636871508), 'PA': (0.15029967727063162, 0.8497003227293684), 'RI':
(0.14903846153846154, 0.8509615384615384), 'SC': (0.1447661469933185,
0.8552338530066815), 'SD': (0.1642512077294686, 0.8357487922705314),
'TN': (0.1493288590604027, 0.8506711409395973), 'TX': (0.1859703218346
5023, 0.8140296781653498), 'UT': (0.16176470588235295, 0.8382352941176
471), 'VA': (0.1503448275862069, 0.8496551724137931), 'VT': (0.2222222
222222222, 0.7777777777777778), 'WA': (0.12515723270440252, 0.87484276
72955975), 'WI': (0.14374514374514374, 0.8562548562548562), 'WV': (0.1
6666666666666666, 0.8333333333333334), 'WY': (0.18840579710144928, 0.8
115942028985508)}
```

Response Coding: teacher_prefix

In [39]:

```
vectorizer_teacher_prefix = ResponseCoding()
vectorizer_teacher_prefix.fit(X_train['clean_teacher_prefix'].values,Y_train.values

X_train_teacher_prefix_response = np.array(vectorizer_teacher_prefix.transform(X_tr
X_test_teacher_prefix_response = np.array(vectorizer_teacher_prefix.transform(X_tes

print(vectorizer_teacher_prefix.get_feature_response_coding())
```

```
{'Dr': (0.4, 0.6), 'Mr': (0.1547049441786284, 0.8452950558213717), 'Mr
s': (0.1450070112179487, 0.8549929887820513), 'Ms': (0.157254815791401
16, 0.8427451842085989), 'Teacher': (0.19244391971664698, 0.8075560802
83353), 'none': (0.0, 1.0)}
```

Response Coding: project_grade_category

In [40]:

```
vectorizer_grade = ResponseCoding()
vectorizer_grade.fit(X_train['clean_grade_category'].values,Y_train.values)


X_train_grade_category_response = np.array(vectorizer_grade.transform(X_train['clea
X_test_grade_category_response = np.array(vectorizer_grade.transform(X_test['clean_

print(vectorizer_grade.get_feature_response_coding())
```

```
{'Grades_3_5': (0.1446957728025895, 0.8553042271974105), 'Grades_6_8':
(0.1584008097165992, 0.8415991902834008), 'Grades_9_12': (0.1629658485
9109206, 0.8370341514089079), 'Grades_PreK_2': (0.15149362183109963,
0.8485063781689004)}
```

2.2.2 Encoding Numerical features

Normalizing Price

In [41]:

```
price_vectorizer = preprocessing.Normalizer().fit(X_train['price'].values.reshape(1
```

In [42]:

```
X_train_price_normalized = price_vectorizer.transform(X_train['price'].values.resha
X_test_price_normalized = price_vectorizer.transform(X_test['price'].values.reshape
```

In [43]:

```
X_train_price_normalized
```

Out[43]:

```
array([[0.00210854, 0.00084169, 0.00107502, ..., 0.00211053, 0.0005170
3,
        0.00010042]])
```

In [44]:

```
X_test_price_normalized
```

Out[44]:

```
array([[0.00322016, 0.00326135, 0.001289  , ..., 0.00558962, 0.0029732
4,
        0.00807531]])
```

Normalize teacher_number_of_previously_posted_projects

In [45]:

```
project_vectorizer = preprocessing.Normalizer().fit(X_train['teacher_number_of_prev
```

In [46]:

```
X_train_normal_previous_project = project_vectorizer.transform(X_train['teacher_num
X_test_normal_previous_project = project_vectorizer.transform(X_test['teacher_numbe
```

2.3 Make Data Model Ready: encoding eassay, and project_title

2.3.1 Bag of words : Essay

In [47]:

```
# We are considering only the words which appeared in at least 10 documents(rows or
vectorizer_essay_bow = CountVectorizer(min_df=10,ngram_range=(1,2), max_features=50
vectorizer_essay_bow.fit(X_train['clean_essay'])
```

Out[47]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='conten
t',
        lowercase=True, max_df=1.0, max_features=5000, min_df=10,
        ngram_range=(1, 2), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [48]:

```
X_train_essay_bow = vectorizer_essay_bow.transform(X_train['clean_essay'])
X_test_essay_bow = vectorizer_essay_bow.transform(X_test['clean_essay'])

print("Shape of X_train_essay_bow ",X_train_essay_bow.shape)
print("Shape of X_test_essay_bow ",X_test_essay_bow.shape)
```

```
Shape of X_train_essay_bow  (76473, 5000)
Shape of X_test_essay_bow  (32775, 5000)
```

2.3.2 Bag of words : Project Title

In [49]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_title_bow = CountVectorizer(min_df=10,ngram_range=(1,2), max_features=50
vectorizer_title_bow.fit(X_train['clean_title'])
```

Out[49]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='conten
t',
        lowercase=True, max_df=1.0, max_features=5000, min_df=10,
        ngram_range=(1, 2), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [50]:

```
X_train_title_bow = vectorizer_title_bow.transform(X_train['clean_title'])
X_test_title_bow = vectorizer_title_bow.transform(X_test['clean_title'])

print("Shape of X_train_title_bow ",X_train_title_bow.shape)
print("Shape of X_test_title_bow ",X_test_title_bow.shape)
```

```
Shape of X_train_title_bow  (76473, 4901)
Shape of X_test_title_bow  (32775, 4901)
```

2.3.3 TFIDF vectorizer: Essay

In [51]:

```
vectorizer_essay_tfidf = TfidfVectorizer(min_df=10,ngram_range=(1,2), max_features=
vectorizer_essay_tfidf.fit(X_train['clean_essay'])
```

Out[51]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='conten
t',
        lowercase=True, max_df=1.0, max_features=5000, min_df=10,
        ngram_range=(1, 2), norm='l2', preprocessor=None, smooth_idf=T
rue,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=Tru
e,
        vocabulary=None)
```

In [52]:

```
X_train_essay_tfidf = vectorizer_essay_tfidf.transform(X_train['clean_essay'])
X_test_essay_tfidf = vectorizer_essay_tfidf.transform(X_test['clean_essay'])

print("Shape of X_train_essay_tfidf ",X_train_essay_tfidf.shape)
print("Shape of X_test_essay_tfidf ",X_test_essay_tfidf.shape)
```

```
Shape of X_train_essay_tfidf  (76473, 5000)
Shape of X_test_essay_tfidf  (32775, 5000)
```

2.3.4 TFIDF vectorizer: Project title

2.3.4 TFIDF Vectorizer: Project title

In [53]:

```
vectorizer_title_tfidf = TfidfVectorizer(min_df=10,ngram_range=(1,2), max_features=
vectorizer_title_tfidf.fit(X_train['clean_title'])
```

Out[53]:

```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.float64'>, encoding='utf-8', input='conten
t',
        lowercase=True, max_df=1.0, max_features=5000, min_df=10,
        ngram_range=(1, 2), norm='l2', preprocessor=None, smooth_idf=T
rue,
        stop_words=None, strip_accents=None, sublinear_tf=False,
        token_pattern='(?u)\\b\\w\\w+\\b', tokenizer=None, use_idf=Tru
e,
        vocabulary=None)
```

In [54]:

```
X_train_title_tfidf = vectorizer_title_tfidf.transform(X_train['clean_title'])
X_test_title_tfidf = vectorizer_title_tfidf.transform(X_test['clean_title'])

print("Shape of X_train_title_tfidf ",X_train_title_tfidf.shape)
print("Shape of X_test_title_tfidf",X_test_title_tfidf.shape)
```

```
Shape of X_train_title_tfidf  (76473, 4901)
Shape of X_test_title_tfidf (32775, 4901)
```

2.3.5 Using Pretrained Models: Avg W2V : Essay

In [55]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-u
# 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 [56]:

```
# average Word2Vec
def get_avg_w2v(corpus):
    avg_w2v_vectors=[]
    for sentence in tqdm(corpus): # 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)
    return avg_w2v_vectors

X_train_essay_avg_w2v_vectors = get_avg_w2v(X_train['clean_essay'])
X_test_essay_avg_w2v_vectors = get_avg_w2v(X_test['clean_essay'])
```

```
100%|██████████| 76473/76473 [00:25<00:00, 3046.59it/s]
100%|██████████| 32775/32775 [00:10<00:00, 3036.83it/s]
```

In [57]:

```
print("Shape of X_train_essay_avg_w2v_vectors",len(X_train_essay_avg_w2v_vectors),l
print("Shape of X_test_essay_avg_w2v_vectors ",len(X_test_essay_avg_w2v_vectors),le
```

```
Shape of X_train_essay_avg_w2v_vectors 76473 300
Shape of X_test_essay_avg_w2v_vectors  32775 300
```

2.3.6 Using Pretrained Models: Avg W2V : Project Title

In [58]:

```
X_train_title_avg_w2v_vectors = get_avg_w2v(X_train['clean_title'])
X_test_title_avg_w2v_vectors = get_avg_w2v(X_test['clean_title'])
```

```
100%|██████████| 76473/76473 [00:01<00:00, 60525.02it/s]
100%|██████████| 32775/32775 [00:00<00:00, 60220.27it/s]
```

2.3.7 Using Pretrained Models: TFIDF weighted W2V : Essay

In [59]:

```
def get_tfidf_weighted_w2v(corpus,dictionary,tfidf_words):
    tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in thi
    for sentence in tqdm(corpus): # 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 va
                tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split(
                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)
    return tfidf_w2v_vectors
```

In [60]:

```
dictionary = dict(zip(vectorizer_essay_tfidf.get_feature_names(), list(vectorizer_e
tfidf_words = set(vectorizer_essay_tfidf.get_feature_names())

X_train_essay_tfidf_w2v_vectors = get_tfidf_weighted_w2v(X_train['clean_essay'].val
X_test_essay_tfidf_w2v_vectors = get_tfidf_weighted_w2v(X_test['clean_essay'].value
```

```
100%|██████████| 76473/76473 [02:18<00:00, 551.32it/s]
100%|██████████| 32775/32775 [00:59<00:00, 554.51it/s]
```

In [61]:

```
print("Shape of X_train_essay_tfidf_w2v_vectors",len(X_train_essay_tfidf_w2v_vector
print("Shape of X_test_essay_tfidf_w2v_vectors ",len(X_test_essay_tfidf_w2v_vectors
```

```
Shape of X_train_essay_tfidf_w2v_vectors 76473 300
Shape of X_test_essay_tfidf_w2v_vectors  32775 300
```

2.3.7 Using Pretrained Models: TFIDF weighted W2V : Project Title

In [62]:

```
dictionary = dict(zip(vectorizer_title_tfidf.get_feature_names(), list(vectorizer_t
tfidf_words = set(vectorizer_title_tfidf.get_feature_names())

X_train_title_tfidf_w2v_vectors = get_tfidf_weighted_w2v(X_train['clean_title'],dic
X_test_title_tfidf_w2v_vectors = get_tfidf_weighted_w2v(X_test['clean_title'],dicti

print("Shape of X_train_title_tfidf_w2v_vectors",len(X_train_title_tfidf_w2v_vector
print("Shape of X_title_title_tfidf_w2v_vectors ",len(X_test_title_tfidf_w2v_vector
```

```
100%|██████████| 76473/76473 [00:02<00:00, 27323.99it/s]
100%|██████████| 32775/32775 [00:01<00:00, 26300.34it/s]

Shape of X_train_title_tfidf_w2v_vectors 76473 300
Shape of X_title_title_tfidf_w2v_vectors  32775 300
```

2.4 Applying Random Forest

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

2.4.1 Applying Random Forests on BOW, SET 1

In [63]:

```
f1 = X_train_school_state_response#np.asarray(X_train_school_state_response).reshap
f2 = X_train_category_response#np.asarray(X_train_category_response).reshape(-1,1)
f3 = X_train_subcategory_response#np.asarray(X_train_subcategory_response).reshape(
f4 = X_train_grade_category_response#np.asarray(X_train_grade_category_response).re
f5 = X_train_teacher_prefix_response#np.asarray(X_train_teacher_prefix_response).re
f6 = np.array(X_train_price_normalized).reshape(-1,1)
f7 = np.array(X_train_normal_previous_project).reshape(-1,1)

X_train_bow = hstack((f1,f2,f3,f4,f5,f6,f7,X_train_essay_bow,X_train_title_bow))
X_train_bow.shape
```

Out[63]:

```
(76473, 9913)
```

In [64]:

```
f1 = X_test_school_state_response#np.asarray(X_test_school_state_response).reshape(
f2 = X_test_category_response#np.asarray(X_test_category_response).reshape(-1,1)
f3 = X_test_subcategory_response#np.asarray(X_test_subcategory_response).reshape(-1
f4 = X_test_grade_category_response#np.asarray(X_test_grade_category_response).resh
f5 = X_test_teacher_prefix_response#np.asarray(X_test_teacher_prefix_response).resh
f6 = np.array(X_test_price_normalized).reshape(-1,1)
f7 = np.array(X_test_normal_previous_project).reshape(-1,1)

X_test_bow = hstack((f1,f2,f3,f4,f5,f6,f7,X_test_essay_bow,X_test_title_bow))
X_test_bow.shape
```

Out[64]:

```
(32775, 9913)
```

Hyperparameter Tuning

In [65]:

```
def train_cv_scores_for_params(model):
    results = pd.DataFrame.from_dict(model.cv_results_)
    max_depths = []
    n_estimators = []
    mean_cv_scores = []
    mean_train_scores = []
    for p in zip(results['params'], results['mean_test_score'], results['mean_train
        param_dict, score_test, score_train = p
        max_depth,n_estimator = param_dict.values()
        max_depths.append(max_depth)
        n_estimators.append(n_estimator)
        mean_cv_scores.append(score_test)
        mean_train_scores.append(score_train)
    return max_depths, n_estimators, mean_cv_scores, mean_train_scores
```

In [167]:

```
rf = RandomForestClassifier(class_weight='balanced')
tune_parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[5, 10,20,30,5
clf = GridSearchCV(rf, tune_parameters, cv= 3, scoring='roc_auc',n_jobs=-1,verbose=
clf.fit(X_train_bow,Y_train)
```

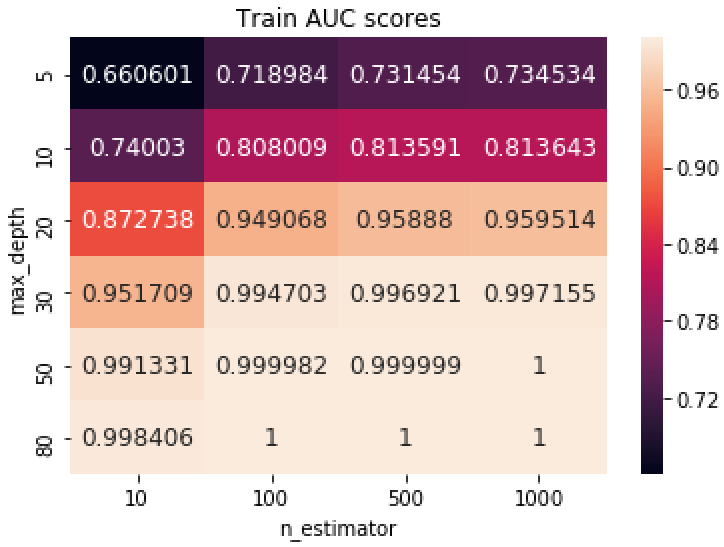
```
Fitting 3 folds for each of 24 candidates, totalling 72 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent wo
rkers.
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:   45.0s
[Parallel(n_jobs=-1)]: Done  72 out of  72 | elapsed: 40.4min finished
```

Out[167]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=RandomForestClassifier(bootstrap=True, class_weight
='balanced',
            criterion='gini', max_depth=None, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators='warn', n_jobs=None, oob_score=False,
            random_state=None, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'n_estimators': [10, 100, 500, 1000], 'max_depth':
[5, 10, 20, 30, 50, 80]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=True)
```

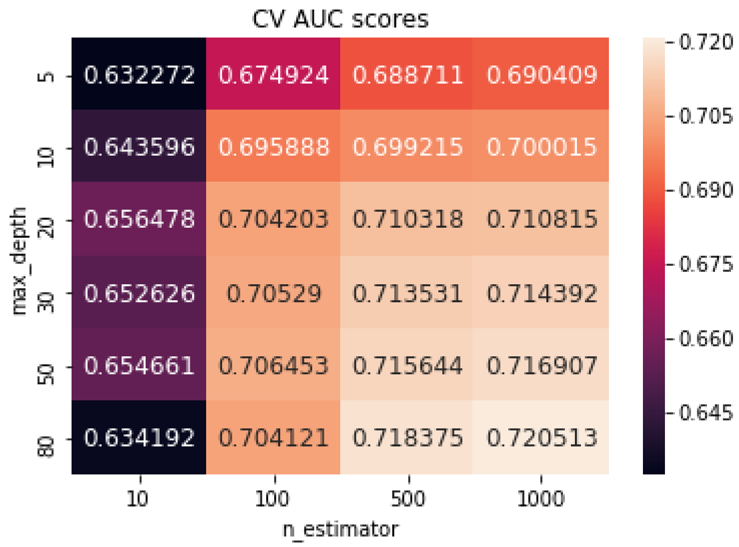
In [168]:

```
max_depths, n_estimators,mean_cv_scores, mean_train_scores = train_cv_scores_for_pa
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("Train AUC scores")
plt.show()
```



In [169]:

```
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("CV AUC scores")
plt.show()
```



In [170]:

```
clf.best_estimator_
```

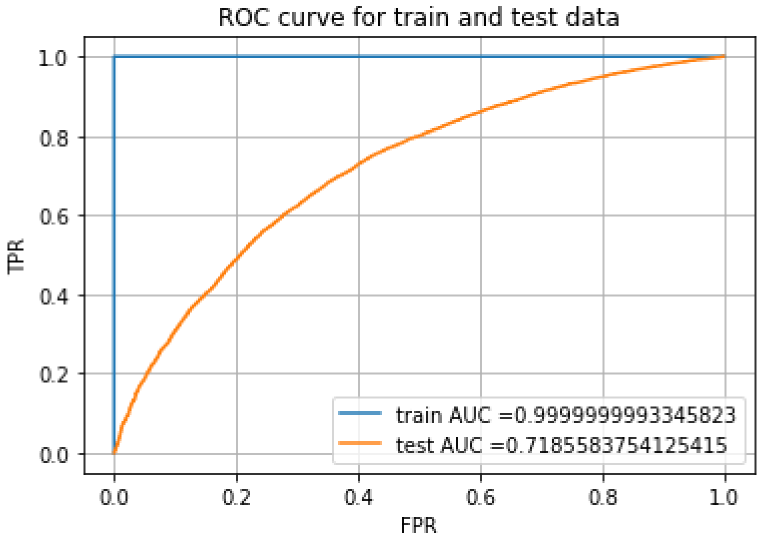
Out[170]:

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=80, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=1000, n_jobs=None, oob_score=False,
            random_state=None, verbose=0, warm_start=False)
```

Training model on optimal value of hyperparameters.

In [171]:

```
rf_bow = clf.best_estimator_#RandomForestClassifier(n_estimators=500,max_depth=10,n
rf_bow.fit(X_train_bow,Y_train)

y_train_pred = rf_bow.predict_proba(X_train_bow)
y_test_pred = rf_bow.predict_proba(X_test_bow)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_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("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



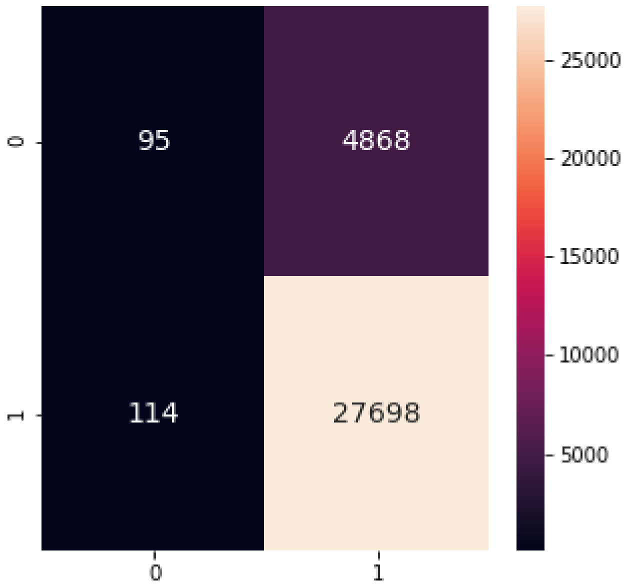
Confusion Matrix

In [172]:

```
y_test_predict = rf_bow.predict(X_test_bow)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[172]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2928a77c88>
```



2.4.2 Applying Random Forests on TFIDF, SET 2

In [65]:

```
f1 = X_train_school_state_response#np.asarray(X_train_school_state_response).reshap
f2 = X_train_category_response#np.asarray(X_train_category_response).reshape(-1,1)
f3 = X_train_subcategory_response#np.asarray(X_train_subcategory_response).reshape(
f4 = X_train_grade_category_response#np.asarray(X_train_grade_category_response).re
f5 = X_train_teacher_prefix_response#np.asarray(X_train_teacher_prefix_response).re
f6 = np.array(X_train_price_normalized).reshape(-1,1)
f7 = np.array(X_train_normal_previous_project).reshape(-1,1)

X_train_tfidf = hstack((f1,f2,f3,f4,f5,f6,f7,X_train_essay_tfidf,X_train_title_tfid
X_train_tfidf.shape
```

Out[65]:

```
(76473, 9826)
```

In [66]:

```
f1 = X_test_school_state_response#np.asarray(X_test_school_state_response).reshape(
f2 = X_test_category_response#np.asarray(X_test_category_response).reshape(-1,1)
f3 = X_test_subcategory_response#np.asarray(X_test_subcategory_response).reshape(-1
f4 = X_test_grade_category_response#np.asarray(X_test_grade_category_response).resh
f5 = X_test_teacher_prefix_response#np.asarray(X_test_teacher_prefix_response).resh
f6 = X_test_price_normalized.reshape(-1,1)
f7 = X_test_normal_previous_project.reshape(-1,1)

X_test_tfidf = hstack((f1,f2,f3,f4,f5,f6,f7,X_test_essay_tfidf,X_test_title_tfidf))
X_test_tfidf.shape
```

Out[66]:

```
(32775, 9826)
```

Hyperparameter Tuning

In [175]:

```
rf = RandomForestClassifier(class_weight='balanced',n_jobs=-1)
tune_parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[5,10,20,30,50
clf = GridSearchCV(rf, tune_parameters, cv= 3, scoring='roc_auc',n_jobs=-1,verbose=
clf.fit(X_train_tfidf,Y_train)
```

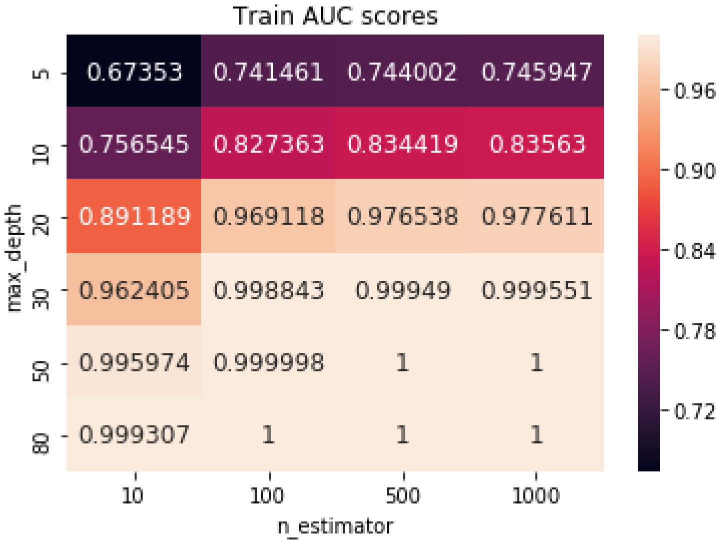
```
Fitting 3 folds for each of 24 candidates, totalling 72 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent wo
rkers.
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:   53.0s
[Parallel(n_jobs=-1)]: Done  72 out of  72 | elapsed: 31.2min finished
```

Out[175]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=RandomForestClassifier(bootstrap=True, class_weight
='balanced',
            criterion='gini', max_depth=None, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators='warn', n_jobs=-1, oob_score=False,
            random_state=None, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'n_estimators': [10, 100, 500, 1000], 'max_depth':
[5, 10, 20, 30, 50, 80]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=True)
```

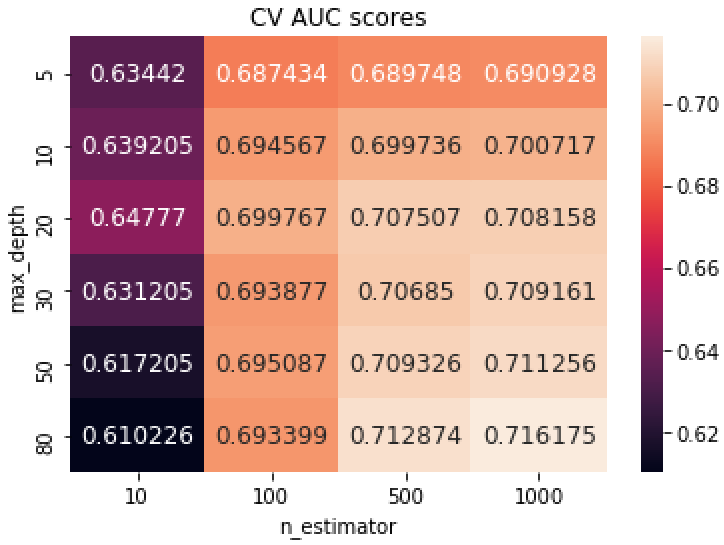
In [176]:

```
max_depths, n_estimators,mean_cv_scores, mean_train_scores = train_cv_scores_for_pa
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("Train AUC scores")
plt.show()
```



In [177]:

```
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("CV AUC scores")
plt.show()
```



In [178]:

```
clf.best_estimator_
```

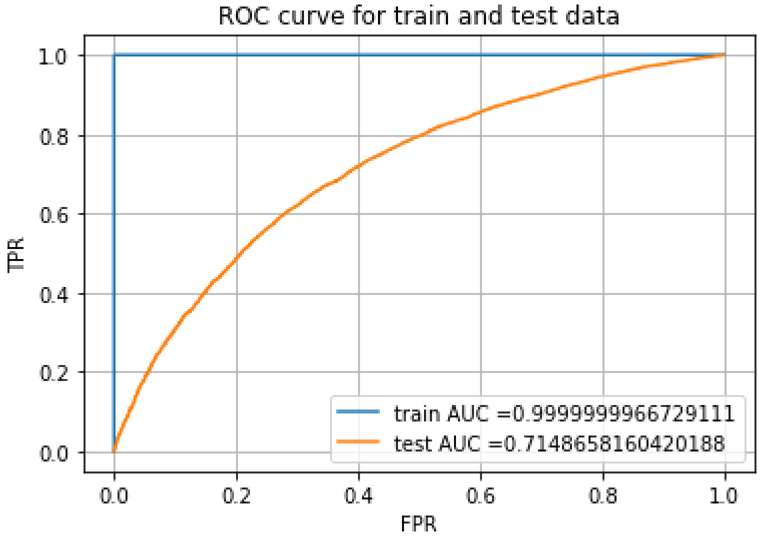
Out[178]:

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=80, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=1000, n_jobs=-1, oob_score=False,
            random_state=None, verbose=0, warm_start=False)
```

Training the model on the most optimal value of hyperparameters

In [179]:

```
rf_tfidf = clf.best_estimator_#RandomForestClassifier(n_estimators=1000,max_depth=1
rf_tfidf.fit(X_train_tfidf,Y_train)

y_train_pred = rf_tfidf.predict_proba(X_train_tfidf)
y_test_pred = rf_tfidf.predict_proba(X_test_tfidf)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_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("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



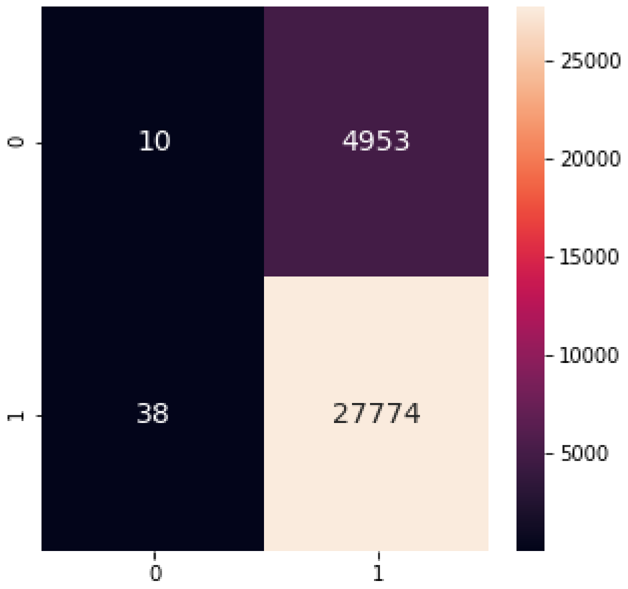
Confusion Matrix

In [180]:

```
y_test_predict = rf_tfidf.predict(X_test_tfidf)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[180]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f288f349048>
```



2.4.3 Applying Random Forests on AVG W2V, SET 3

In [66]:

```
f1 = X_train_school_state_response#np.asarray(X_train_school_state_response).reshap
f2 = X_train_category_response#np.asarray(X_train_category_response).reshape(-1,1)

f3 = X_train_subcategory_response#np.asarray(X_train_subcategory_response).reshape(
f4 = X_train_grade_category_response#np.asarray(X_train_grade_category_response).re
f5 = X_train_teacher_prefix_response#np.asarray(X_train_teacher_prefix_response).re
f6 = np.array(X_train_price_normalized).reshape(-1,1)
f7 = np.array(X_train_normal_previous_project).reshape(-1,1)

f8 = X_train_essay_avg_w2v_vectors
f9 = X_train_title_avg_w2v_vectors
X_train_w2v = np.hstack((f1,f2,f3,f4,f5,f6,f7,f8,f9))
X_train_w2v.shape
```

Out[66]:

```
(76473, 612)
```

In [67]:

```
f1 = X_test_school_state_response#np.asarray(X_test_school_state_response).reshape(
f2 = X_test_category_response#np.asarray(X_test_category_response).reshape(-1,1)
f3 = X_test_subcategory_response#np.asarray(X_test_subcategory_response).reshape(-1
f4 = X_test_grade_category_response#np.asarray(X_test_grade_category_response).resh
f5 = X_test_teacher_prefix_response#np.asarray(X_test_teacher_prefix_response).resh
f6 = X_test_price_normalized.reshape(-1,1).reshape(-1,1)
f7 = X_test_normal_previous_project.reshape(-1,1)

X_test_w2v = np.hstack((f1,f2,f3,f4,f5,f6,f7,X_test_essay_avg_w2v_vectors,X_test_ti
X_test_w2v.shape
```

Out[67]:

```
(32775, 612)
```

Hyperparameter Tuning

In [183]:

```
rf = RandomForestClassifier(class_weight='balanced',n_jobs=-1)
tune_parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[5, 10,20,30,5
clf = GridSearchCV(rf, tune_parameters, cv= 3, scoring='roc_auc',n_jobs=-1,verbose=
clf.fit(X_train_w2v,Y_train)
```

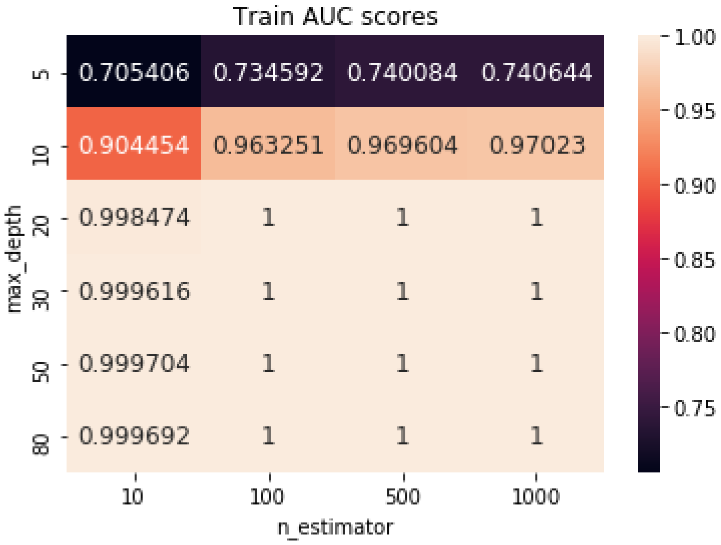
```
Fitting 3 folds for each of 24 candidates, totalling 72 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent wo
rkers.
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:  8.4min
[Parallel(n_jobs=-1)]: Done  72 out of  72 | elapsed: 63.3min finished
```

Out[183]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=RandomForestClassifier(bootstrap=True, class_weight
='balanced',
            criterion='gini', max_depth=None, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators='warn', n_jobs=-1, oob_score=False,
            random_state=None, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'n_estimators': [10, 100, 500, 1000], 'max_depth':
[5, 10, 20, 30, 50, 80]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=True)
```

In [184]:

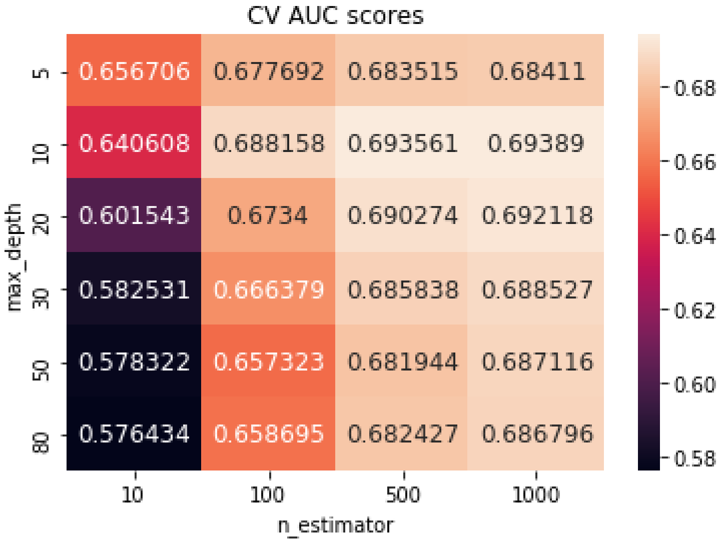
```
max_depths, n_estimators,mean_cv_scores, mean_train_scores = train_cv_scores_for_pa

df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("Train AUC scores")
plt.show()
```



max_depth	10	100	500	1000
5	0.705406	0.734592	0.740084	0.740644
10	0.904454	0.963251	0.969604	0.97023
20	0.998474	1	1	1
30	0.999616	1	1	1
50	0.999704	1	1	1
80	0.999692	1	1	1

In [185]:

```
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("CV AUC scores")
plt.show()
```



In [186]:

```
clf.best_estimator_
```

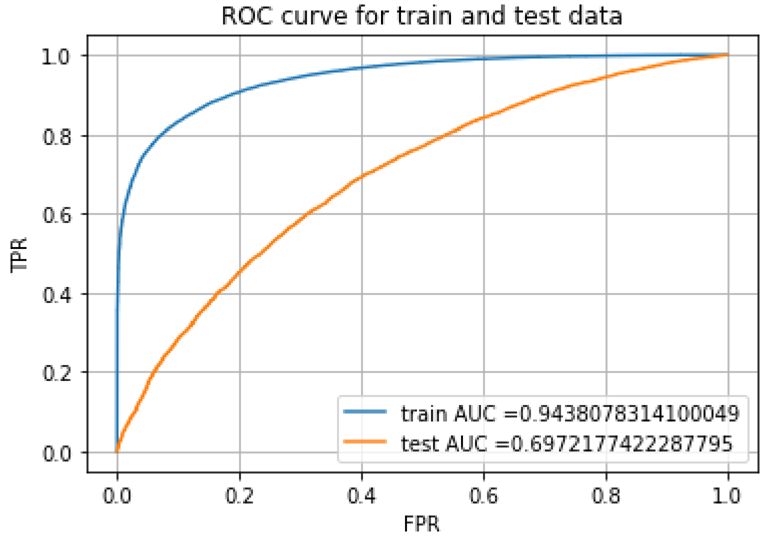
Out[186]:

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=10, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=1000, n_jobs=-1, oob_score=False,
            random_state=None, verbose=0, warm_start=False)
```

Training the model on the most optimal value of hyperparameters

In [187]:

```
rf_w2v = clf.best_estimator_#RandomForestClassifier(n_estimators=1000,max_depth=10,
rf_w2v.fit(X_train_w2v,Y_train)

y_train_pred = rf_w2v.predict_proba(X_train_w2v)
y_test_pred = rf_w2v.predict_proba(X_test_w2v)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_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("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



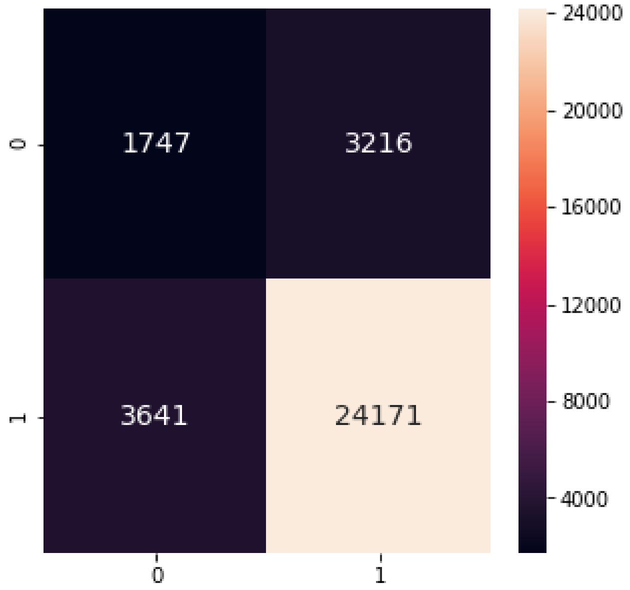
Confusion Matrix

In [188]:

```
y_test_predict = rf_w2v.predict(X_test_w2v)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[188]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f291ef60198>
```



2.4.4 Applying Random Forests on TFIDF W2V, SET 4

In [68]:

```
f1 = X_train_school_state_response#np.asarray(X_train_school_state_response).reshap
f2 = X_train_category_response#np.asarray(X_train_category_response).reshape(-1,1)
f3 = X_train_subcategory_response#np.asarray(X_train_subcategory_response).reshape(
f4 = X_train_grade_category_response#np.asarray(X_train_grade_category_response).re
f5 = X_train_teacher_prefix_response#np.asarray(X_train_teacher_prefix_response).re
f6 = np.array(X_train_price_normalized).reshape(-1,1)
f7 = np.array(X_train_normal_previous_project).reshape(-1,1)

X_train_tfidf_w2v = np.hstack((f1,f2,f3,f4,f5,f6,f7,X_train_essay_tfidf_w2v_vectors
X_train_tfidf_w2v.shape
```

Out[68]:

```
(76473, 612)
```

In [69]:

```
f1 = X_test_school_state_response#np.asarray(X_test_school_state_response).reshape(
f2 = X_test_category_response#np.asarray(X_test_category_response).reshape(-1,1)
f3 = X_test_subcategory_response#np.asarray(X_test_subcategory_response).reshape(-1
f4 = X_test_grade_category_response#np.asarray(X_test_grade_category_response).resh
f5 = X_test_teacher_prefix_response#np.asarray(X_test_teacher_prefix_response).resh
f6 = X_test_price_normalized.reshape(-1,1)
f7 = X_test_normal_previous_project.reshape(-1,1)

X_test_tfidf_w2v = np.hstack((f1,f2,f3,f4,f5,f6,f7,X_test_essay_tfidf_w2v_vectors,X
X_test_tfidf_w2v.shape
```

Out[69]:

```
(32775, 612)
```

In [191]:

```
rf = RandomForestClassifier(class_weight='balanced',n_jobs=-1)
tune_parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[5,10,20,30,50
clf = GridSearchCV(rf, tune_parameters, cv= 3, scoring='roc_auc',n_jobs=-1,verbose=
clf.fit(X_train_tfidf_w2v,Y_train)
```

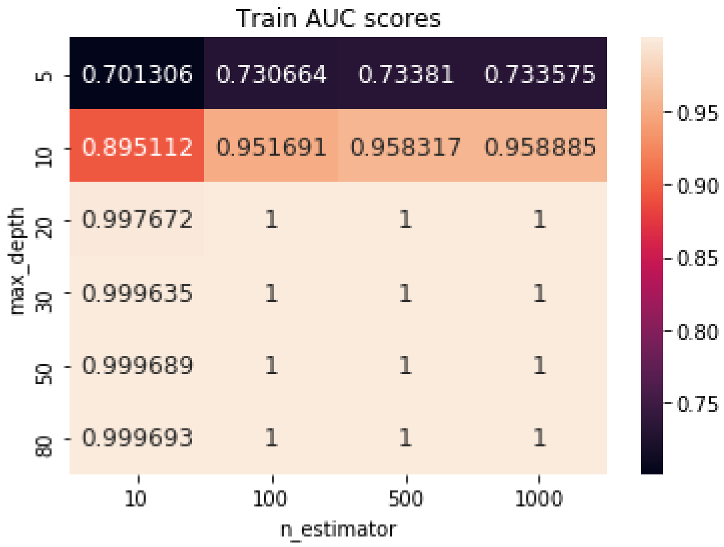
```
Fitting 3 folds for each of 24 candidates, totalling 72 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent wo
rkers.
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:  8.3min
[Parallel(n_jobs=-1)]: Done  72 out of  72 | elapsed: 62.9min finished
```

Out[191]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=RandomForestClassifier(bootstrap=True, class_weight
='balanced',
            criterion='gini', max_depth=None, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators='warn', n_jobs=-1, oob_score=False,
            random_state=None, verbose=0, warm_start=False),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'n_estimators': [10, 100, 500, 1000], 'max_depth':
[5, 10, 20, 30, 50, 80]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=True)
```

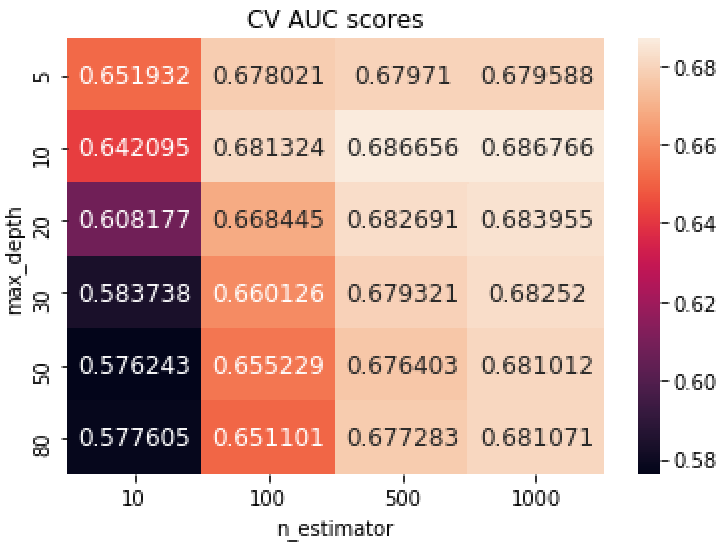
In [192]:

```
max_depths, n_estimators,mean_cv_scores, mean_train_scores = train_cv_scores_for_pa

df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("Train AUC scores")
plt.show()
```



In [193]:

```
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("CV AUC scores")
plt.show()
```



In [194]:

```
clf.best_estimator_
```

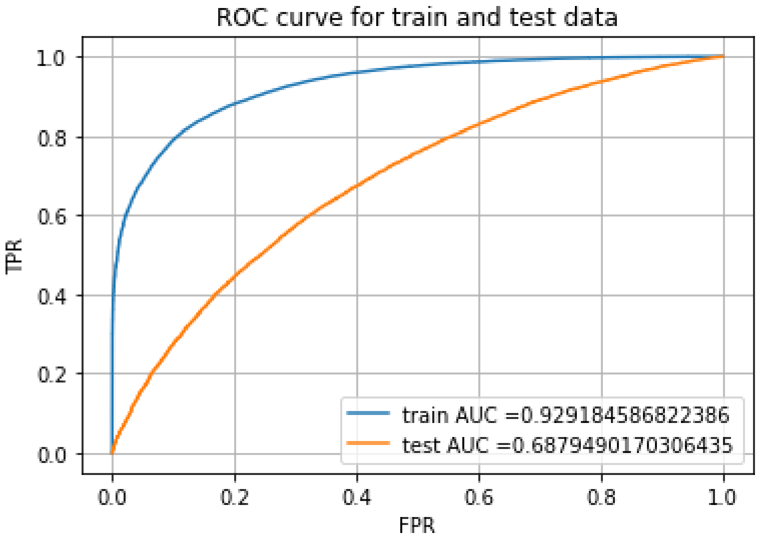
Out[194]:

```
RandomForestClassifier(bootstrap=True, class_weight='balanced',
            criterion='gini', max_depth=10, max_features='auto',
            max_leaf_nodes=None, min_impurity_decrease=0.0,
            min_impurity_split=None, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=1000, n_jobs=-1, oob_score=False,
            random_state=None, verbose=0, warm_start=False)
```

Training model on optimal values of hyperparameters

In [195]:

```
rf_tfidf_w2v = clf.best_estimator_#RandomForestClassifier(n_estimators=1000,max_dep
rf_tfidf_w2v.fit(X_train_tfidf_w2v,Y_train)

y_train_pred = rf_tfidf_w2v.predict_proba(X_train_tfidf_w2v)
y_test_pred = rf_tfidf_w2v.predict_proba(X_test_tfidf_w2v)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_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("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```



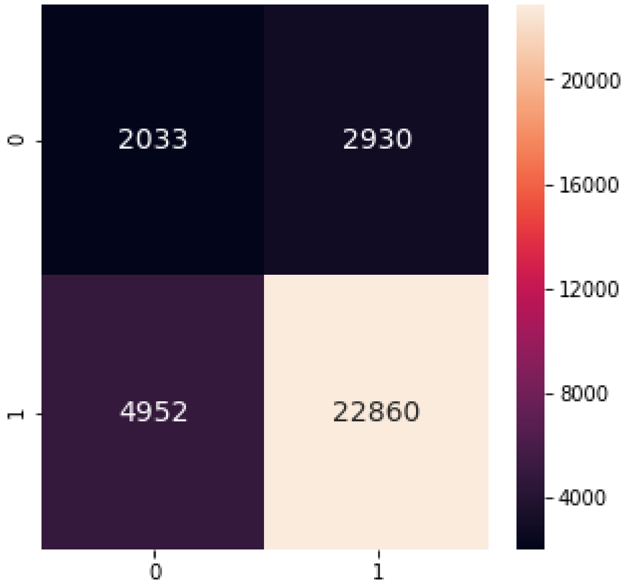
Confusion Matrix

In [196]:

```
y_test_predict = rf_tfidf_w2v.predict(X_test_tfidf_w2v)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[196]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f292d14b518>
```



2.5 Applying GBDT

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

2.5.1 Applying XGBOOST on BOW, SET 1

In [71]:

```
xgb = XGBClassifier(class_weight='balanced')
tune_parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[2, 3, 4, 5, 6
clf = GridSearchCV(xgb, tune_parameters, cv= 3, scoring='roc_auc',n_jobs=-1,verbose
clf.fit(X_train_bow,Y_train)
```

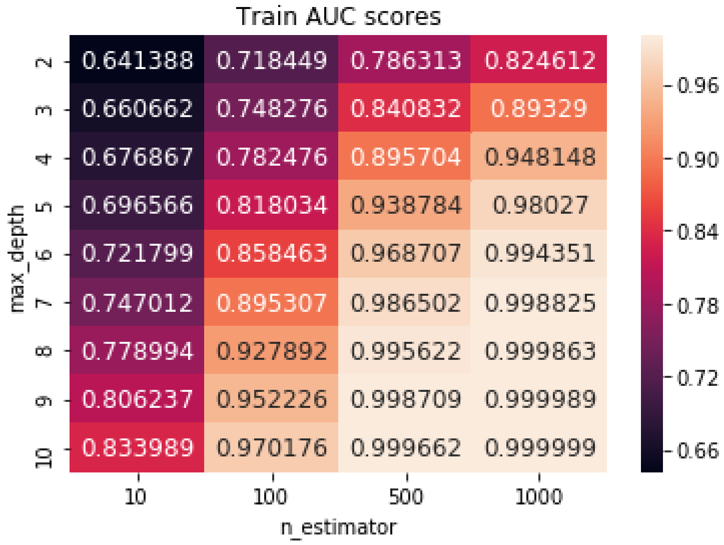
```
Fitting 3 folds for each of 36 candidates, totalling 108 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent wo
rkers.
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:  2.5min
[Parallel(n_jobs=-1)]: Done 108 out of 108 | elapsed: 47.2min finished
```

Out[71]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=XGBClassifier(base_score=0.5, booster='gbtree', class
_weight='balanced',
       colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
       gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=3,
       min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
       nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'n_estimators': [10, 100, 500, 1000], 'max_depth':
[2, 3, 4, 5, 6, 7, 8, 9, 10]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=True)
```

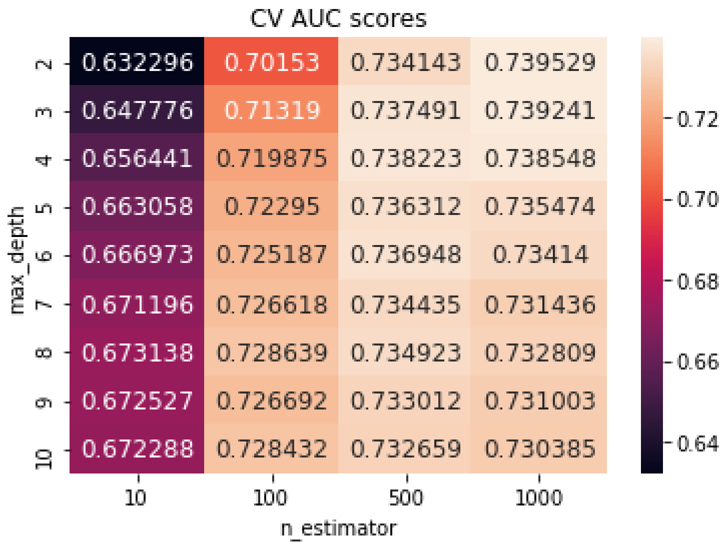
In [74]:

```
max_depths, n_estimators,mean_cv_scores, mean_train_scores = train_cv_scores_for_pa

df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("Train AUC scores")
plt.show()
```



In [75]:

```
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("CV AUC scores")
plt.show()
```



In [76]:

```
clf.best_estimator_
```

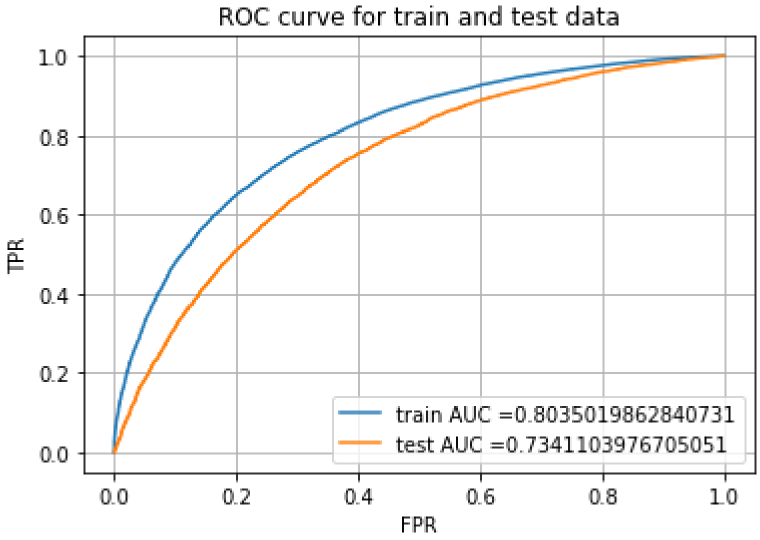
Out[76]:

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balance
d',
       colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
       gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=2,
       min_child_weight=1, missing=None, n_estimators=1000, n_jobs=1,
       nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1)
```

Tuning the model on the best hyperparameters

In [77]:

```
xgb_bow = clf.best_estimator_#XGBClassifier(n_estimators=500,max_depth=10,class_wei
xgb_bow.fit(X_train_bow,Y_train)

y_train_pred = xgb_bow.predict_proba(X_train_bow)
y_test_pred = xgb_bow.predict_proba(X_test_bow)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_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("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```

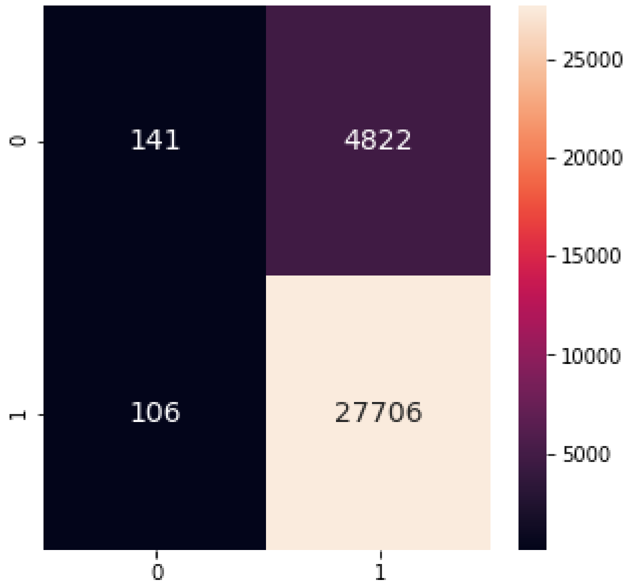


In [78]:

```
y_test_predict = xgb_bow.predict(X_test_bow)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[78]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1bd22a1eb8>
```



2.5.2 Applying XGBOOST on TFIDF, SET 2

In [79]:

```
gbdt = XGBClassifier()
tune_parameters = {'n_estimators': [10, 100, 500, 1000], 'max_depth':[2, 3, 4, 5, 6
clf = GridSearchCV(gbdt, tune_parameters, cv= 3, scoring='roc_auc',n_jobs=-1,verbos
clf.fit(X_train_tfidf,Y_train)
```

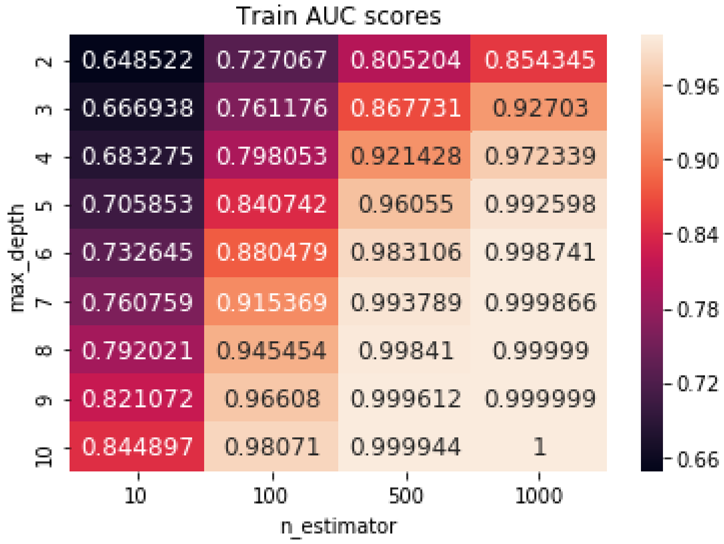
```
Fitting 3 folds for each of 36 candidates, totalling 108 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent wo
rkers.
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:  6.4min
[Parallel(n_jobs=-1)]: Done 108 out of 108 | elapsed: 140.0min finishe
d
```

Out[79]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=XGBClassifier(base_score=0.5, booster='gbtree', colsa
mple_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=
0.1,
       max_delta_step=0, max_depth=3, min_child_weight=1, missing=Non
e,
       n_estimators=100, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'n_estimators': [10, 100, 500, 1000], 'max_depth':
[2, 3, 4, 5, 6, 7, 8, 9, 10]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=True)
```

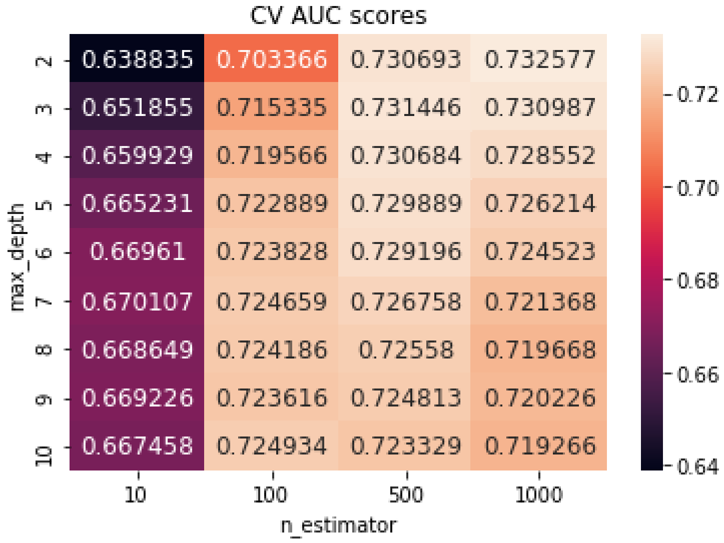
In [80]:

```
max_depths, n_estimators,mean_cv_scores, mean_train_scores = train_cv_scores_for_pa

df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("Train AUC scores")
plt.show()
```



In [81]:

```
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("CV AUC scores")
plt.show()
```



In [82]:

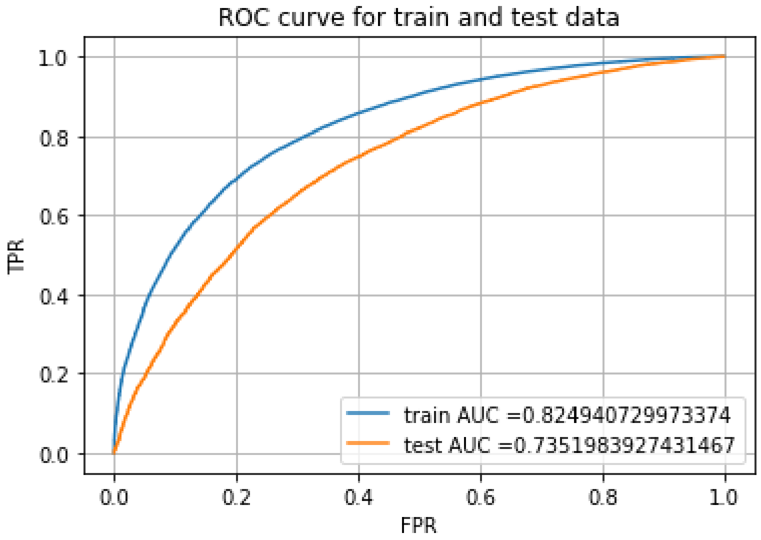
```
clf.best_estimator_
```

Out[82]:

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bynode=1, colsample_bytree=1, gamma=0, learning_rate=
0.1,
       max_delta_step=0, max_depth=2, min_child_weight=1, missing=Non
e,
       n_estimators=1000, n_jobs=1, nthread=None,
       objective='binary:logistic', random_state=0, reg_alpha=0,
       reg_lambda=1, scale_pos_weight=1, seed=None, silent=None,
       subsample=1, verbosity=1)
```

In [83]:

```
xgb_tfidf = clf.best_estimator_#XGBClassifier(n_estimators=500,max_depth=50,class_w
xgb_tfidf.fit(X_train_tfidf,Y_train)

y_train_pred = xgb_tfidf.predict_proba(X_train_tfidf)
y_test_pred = xgb_tfidf.predict_proba(X_test_tfidf)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_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("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```

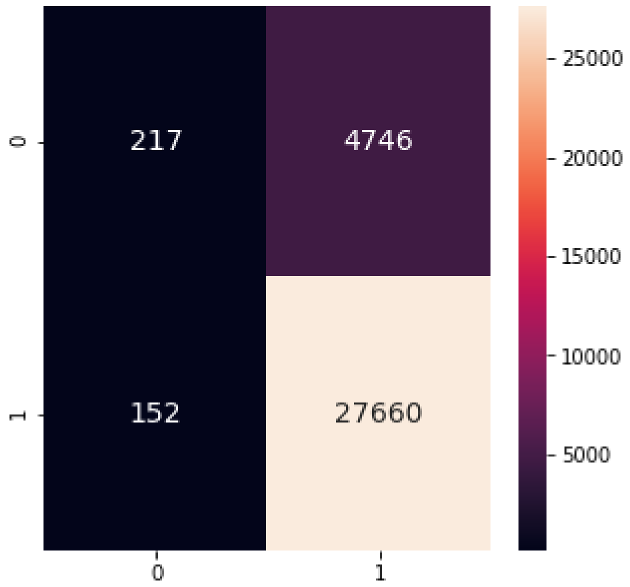


In [84]:

```
y_test_predict = xgb_tfidf.predict(X_test_tfidf)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[84]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1bc9f40be0>
```



2.5.3 Applying XGBOOST on AVG W2V, SET 3

In [71]:

```
gbdt = XGBClassifier(class_weight='balanced')
tune_parameters = {'n_estimators': [10, 100, 500], 'max_depth':[2, 3, 4, 5, 6, 7, 8
clf = GridSearchCV(gbdt, tune_parameters, cv= 3, scoring='roc_auc',n_jobs=-1,verbos
clf.fit(X_train_w2v,Y_train)
```

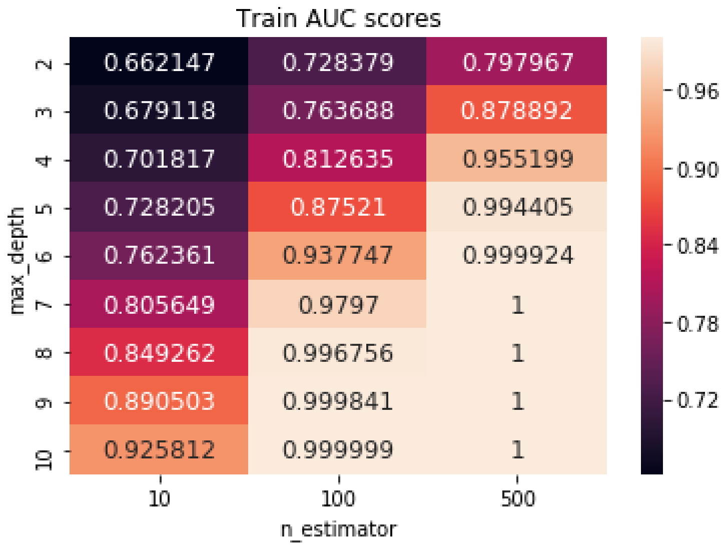
```
Fitting 3 folds for each of 27 candidates, totalling 81 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent wo
rkers.
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:  7.3min
[Parallel(n_jobs=-1)]: Done  81 out of  81 | elapsed: 148.8min finishe
d
```

Out[71]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=XGBClassifier(base_score=0.5, booster='gbtree', class
_weight='balanced',
       colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
       gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=3,
       min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
       nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'n_estimators': [10, 100, 500], 'max_depth': [2, 3,
4, 5, 6, 7, 8, 9, 10]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=True)
```

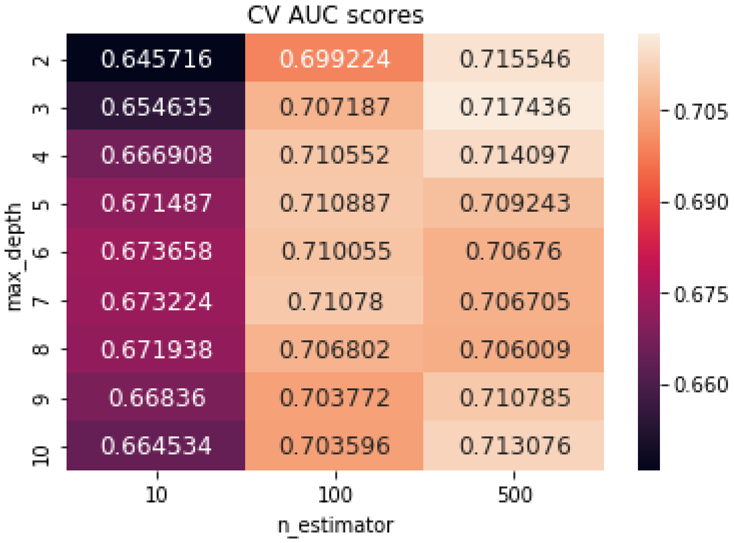
In [72]:

```
max_depths, n_estimators,mean_cv_scores, mean_train_scores = train_cv_scores_for_pa

df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("Train AUC scores")
plt.show()
```



In [73]:

```
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("CV AUC scores")
plt.show()
```



In [74]:

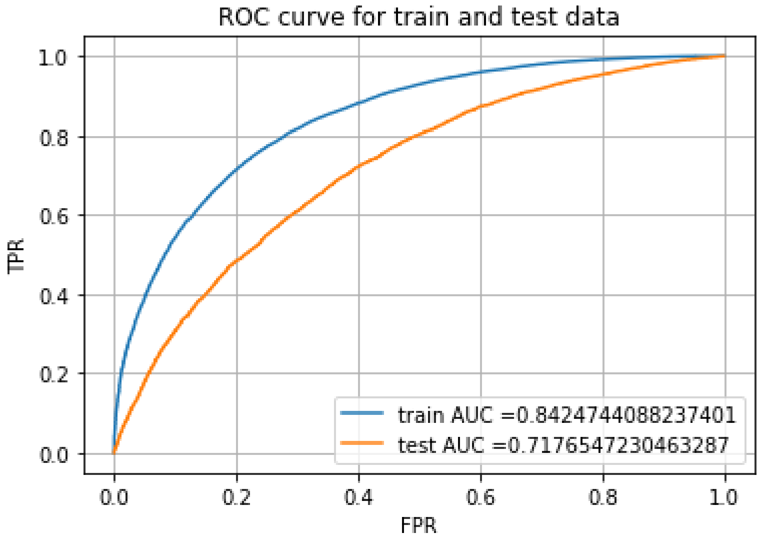
```
clf.best_estimator_
```

Out[74]:

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balance
d',
       colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
       gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=3,
       min_child_weight=1, missing=None, n_estimators=500, n_jobs=1,
       nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1)
```

In [75]:

```
xgb_w2v = clf.best_estimator_#XGBClassifier(n_estimators=500,max_depth=3,class_weig
xgb_w2v.fit(X_train_w2v,Y_train)

y_train_pred = xgb_w2v.predict_proba(X_train_w2v)
y_test_pred = xgb_w2v.predict_proba(X_test_w2v)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_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("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```

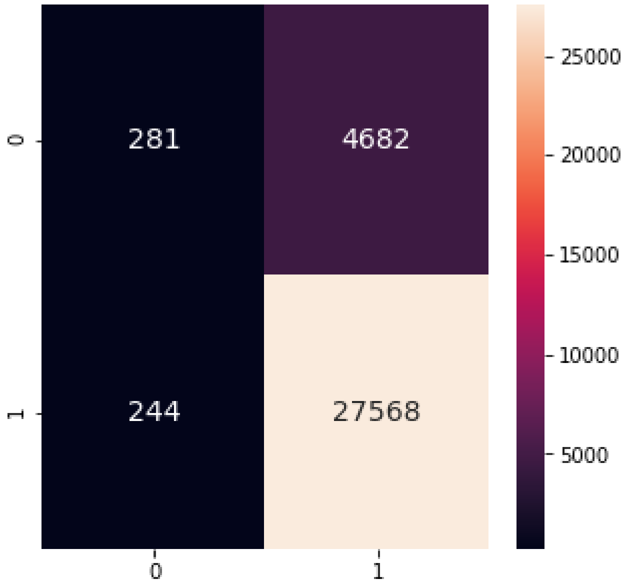


In [76]:

```
y_test_predict = xgb_w2v.predict(X_test_w2v)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[76]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa8e6ad98d0>
```



2.5.4 Applying XGBOOST on TFIDF W2V, SET 4

In [77]:

```
gbdt = XGBClassifier(class_weight='balanced')
tune_parameters = {'n_estimators': [10, 100, 500], 'max_depth':[2, 3, 4, 5, 6, 7, 8
clf = GridSearchCV(gbdt, tune_parameters, cv= 3, scoring='roc_auc',n_jobs=-1,verbos
clf.fit(X_train_tfidf_w2v,Y_train)
```

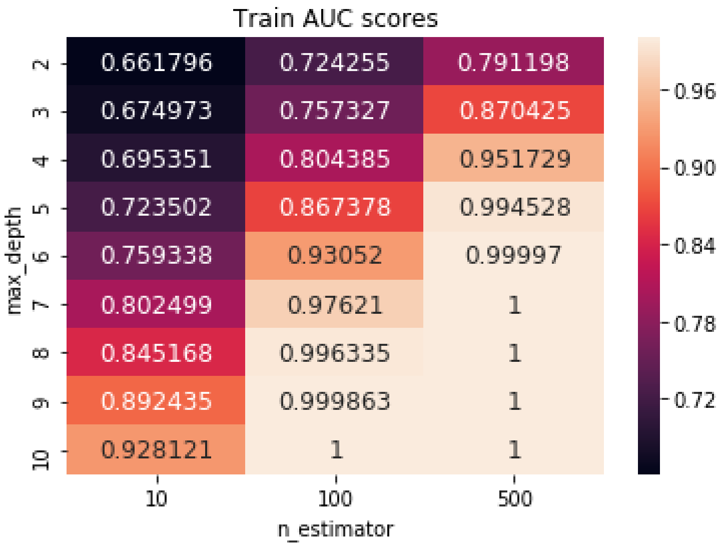
```
Fitting 3 folds for each of 27 candidates, totalling 81 fits

[Parallel(n_jobs=-1)]: Using backend LokyBackend with 16 concurrent wo
rkers.
[Parallel(n_jobs=-1)]: Done  18 tasks      | elapsed:  7.2min
[Parallel(n_jobs=-1)]: Done  81 out of  81 | elapsed: 147.4min finishe
d
```

Out[77]:

```
GridSearchCV(cv=3, error_score='raise-deprecating',
       estimator=XGBClassifier(base_score=0.5, booster='gbtree', class
_weight='balanced',
       colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
       gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=3,
       min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
       nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1),
       fit_params=None, iid='warn', n_jobs=-1,
       param_grid={'n_estimators': [10, 100, 500], 'max_depth': [2, 3,
4, 5, 6, 7, 8, 9, 10]},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring='roc_auc', verbose=True)
```

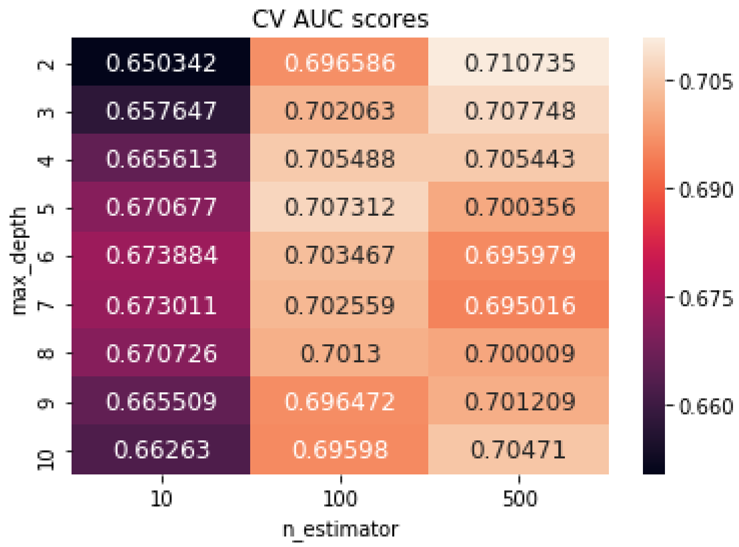
In [78]:

```
max_depths, n_estimators,mean_cv_scores, mean_train_scores = train_cv_scores_for_pa

df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("Train AUC scores")
plt.show()
```



In [79]:

```
df = pd.DataFrame({'max_depth':max_depths,'n_estimator':n_estimators,'mean_test_sco
pivot = df.pivot(index = "max_depth", columns = "n_estimator", values="mean_test_sc
sns.heatmap(pivot,annot=True, annot_kws={"size": 12}, fmt='g')
plt.title("CV AUC scores")
plt.show()
```



In [80]:

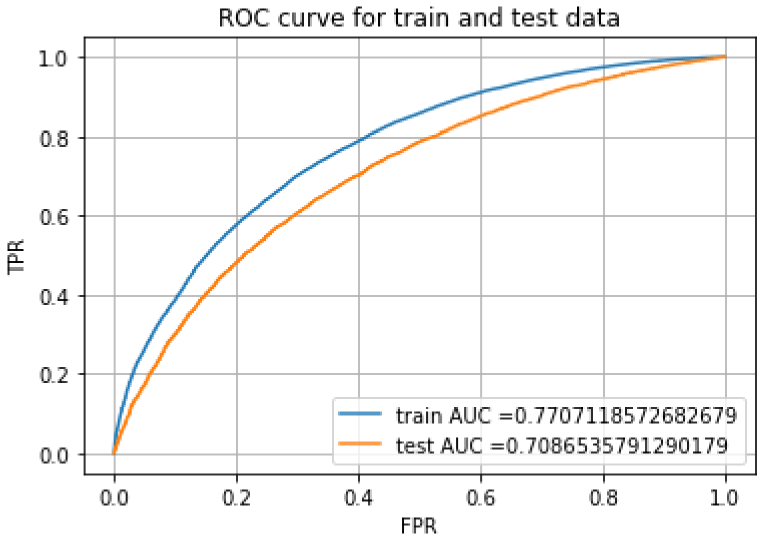
```
clf.best_estimator_
```

Out[80]:

```
XGBClassifier(base_score=0.5, booster='gbtree', class_weight='balance
d',
       colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
       gamma=0, learning_rate=0.1, max_delta_step=0, max_depth=2,
       min_child_weight=1, missing=None, n_estimators=500, n_jobs=1,
       nthread=None, objective='binary:logistic', random_state=0,
       reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
       silent=None, subsample=1, verbosity=1)
```

In [81]:

```
xgb_tfidf_w2v = clf.best_estimator_#XGBClassifier(n_estimators=500,max_depth=3,clas
xgb_tfidf_w2v.fit(X_train_tfidf_w2v,Y_train)

y_train_pred = xgb_tfidf_w2v.predict_proba(X_train_tfidf_w2v)
y_test_pred = xgb_tfidf_w2v.predict_proba(X_test_tfidf_w2v)

# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
train_fpr, train_tpr, tr_thresholds = roc_curve(Y_train, y_train_pred[:,1])
test_fpr, test_tpr, te_thresholds = roc_curve(Y_test, y_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("FPR")
plt.ylabel("TPR")
plt.title("ROC curve for train and test data")
plt.grid()
plt.show()
```

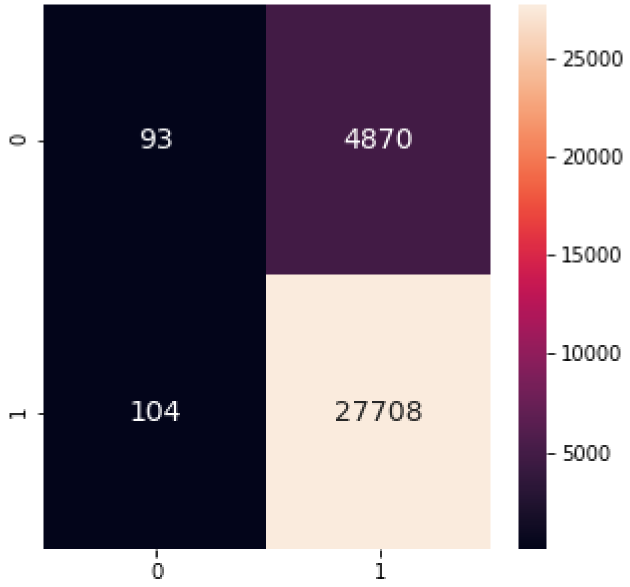


In [82]:

```
y_test_predict = xgb_tfidf_w2v.predict(X_test_tfidf_w2v)

results = confusion_matrix(Y_test, y_test_predict)
plt.figure(figsize = (5,5))
sns.heatmap(results, annot=True,annot_kws={"size": 14}, fmt='g')
```

Out[82]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa8d0ca0470>
```



3. Conclusion

In [83]:

```
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "n_estimator:hyperparam","max_depth:hyperpa

x.add_row(["BOW", "Ramdom Forest",1000,80,0.718])
x.add_row(["TFIDF", "Ramdom Forest", 1000,80,0.714])
x.add_row(["W2Vec", "Ramdom Forest", 1000,10,0.697])
x.add_row(["TFIDF-W2Vec", "Ramdom Forest", 1000, 10,0.687])
x.add_row(["BOW", "GBDT", 1000,2,0.7341])
x.add_row(["TFIDF", "GBDT", 1000,2,0.735])
x.add_row(["W2Vec", "GBDT", 500,3,0.717])
x.add_row(["TFIDF-W2Vec", "GBDT", 500,2,0.708])
print(x)
```

```
+-------------+---------------+------------------------+--------------
--------+--------+
|  Vectorizer |     Model     | n_estimator:hyperparam | max_depth:hyp
erparam |  AUC   |
+-------------+---------------+------------------------+--------------
--------+--------+
|     BOW     | Ramdom Forest |          1000          |          80
| 0.718  |
|    TFIDF    | Ramdom Forest |          1000          |          80
| 0.714  |
|    W2Vec    | Ramdom Forest |          1000          |          10
| 0.697  |
| TFIDF-W2Vec | Ramdom Forest |          1000          |          10
| 0.687  |
|     BOW     |      GBDT     |          1000          |          2
| 0.7341 |
|    TFIDF    |      GBDT     |          1000          |          2
| 0.735  |
|    W2Vec    |      GBDT     |          500           |          3
| 0.717  |
| TFIDF-W2Vec |      GBDT     |          500           |          2
| 0.708  |
+-------------+---------------+------------------------+--------------
--------+--------+
```

Present Slides Themes Help