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	Descri
`project_id`	A unique identifier for the proposed project. **Example:** `p03`
`project_title`	Title of the project. **Exam** • `Art Will Make You Ha` • `First Grade`
`project_grade_category`	Grade level of students for which the project is targeted. One of the follo enumerated va • `Grades Pr` • `Grades` • `Grades` • `Grades`

Feature	Descri
project_subject_categories	One or more (comma-separated) subject categories for the project fro following enumerated list of va • Applied Lear • Care & Hu • Health & Sp • History & Ci • Literacy & Lang • Math & Sci • Music & The • Special N • Wa **Exam** • Music & The • Literacy & Language, Math & Sci
school_state	State where school is located (Two-letter U.S. postal (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_co **Exampl**
project_subject_subcategories	One or more (comma-separated) subject subcategories for the pr **Exam** • Lite • Literature & Writing, Social Scie
project_resource_summary	An explanation of the resources needed for the project. **Exan** • My students need hands on literacy materials to mar sensory ne
project_essay_1	First application e
project_essay_2	Second application e
project_essay_3	Third application e
project_essay_4	Fourth application e
project_submitted_datetime	Datetime when project application was submitted. **Example:** 2016-04 12:43:56
teacher_id	A unique identifier for the teacher of the proposed project. **Exa** bdf8baa8fedef6bfeec7ae4ff1c1
teacher_prefix	Teacher's title. One of the following enumerated va • • • • • • Teac
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same tea **Exam**

* 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
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.
```

1.1 Reading Data

In [2]:

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

In [3]:

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 's
chool_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

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

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[4]:

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

1.2 preprocessing of `project_subject_categories`

In [5]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

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

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

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

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

1.3 preprocessing of project_subject_subcategories

In [6]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

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

sub_cat_list = []
for i in sub_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", "Warmth", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

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

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

In [ ]:

```

```

1.3 Text preprocessing

In [7]:

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

In [8]:

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

Out[8]:

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

In [9]:

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

In [10]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second
or third languages. We are a melting pot of refugees, immigrants, and native
-born Americans bringing the gift of language to our school. \r\n\r\n We hav
e over 24 languages represented in our English Learner program with students
at every level of mastery.  We also have over 40 countries represented with
the families within our school.  Each student brings a wealth of knowledge a
nd experiences to us that open our eyes to new cultures, beliefs, and respec
t.\"The limits of your language are the limits of your world.\"-Ludwig Wittg
enstein  Our English learner's have a strong support system at home that beg
s for more resources.  Many times our parents are learning to read and speak
English along side of their children.  Sometimes this creates barriers for p
arents to be able to help their child learn phonetics, letter recognition, a
nd other reading skills.\r\n\r\nBy providing these dvd's and players, studen
ts are able to continue their mastery of the English language even if no one
at home is able to assist.  All families with students within the Level 1 pr
oficiency status, will be a offered to be a part of this program.  These edu
cational videos will be specially chosen by the English Learner Teacher and
will be sent home regularly to watch.  The videos are to help the child deve
lop early reading skills.\r\n\r\nParents that do not have access to a dvd pl
ayer will have the opportunity to check out a dvd player to use for the yea
r.  The plan is to use these videos and educational dvd's for the years to c
ome for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year a
ll love learning, at least most of the time. At our school, 97.3% of the stu
dents receive free or reduced price lunch. Of the 560 students, 97.3% are mi
nority students. \r\nThe school has a vibrant community that loves to get to
gether and celebrate. Around Halloween there is a whole school parade to sho
w off the beautiful costumes that students wear. On Cinco de Mayo we put on
a big festival with crafts made by the students, dances, and games. At the e
nd of the year the school hosts a carnival to celebrate the hard work put in
during the school year, with a dunk tank being the most popular activity.My
students will use these five brightly colored Hokki stools in place of regul
ar, stationary, 4-legged chairs. As I will only have a total of ten in the c
lassroom and not enough for each student to have an individual one, they wil
l be used in a variety of ways. During independent reading time they will be
used as special chairs students will each use on occasion. I will utilize th
em in place of chairs at my small group tables during math and reading time
s. The rest of the day they will be used by the students who need the highes
t amount of movement in their life in order to stay focused on school.\r\n\r
\nWhenever asked what the classroom is missing, my students always say more
Hokki Stools. They can't get their fill of the 5 stools we already have. Whe
n the students are sitting in group with me on the Hokki Stools, they are al
ways moving, but at the same time doing their work. Anytime the students get
to pick where they can sit, the Hokki Stools are the first to be taken. Ther
e are always students who head over to the kidney table to get one of the st
ools who are disappointed as there are not enough of them. \r\n\r\nWe ask a
```

lot of students to sit for 7 hours a day. The Hokki stools will be a comprom
ise that allow my students to do desk work and move at the same time. These
stools will help students to meet their 60 minutes a day of movement by allo
wing them to activate their core muscles for balance while they sit. For man
y of my students, these chairs will take away the barrier that exists in sch
ools for a child who can't sit still.nannan
==========
How do you remember your days of school? Was it in a sterile environment wit
h plain walls, rows of desks, and a teacher in front of the room? A typical
day in our room is nothing like that. I work hard to create a warm inviting
themed room for my students look forward to coming to each day.\r\n\r\nMy cl
ass is made up of 28 wonderfully unique boys and girls of mixed races in Ark
ansas.\r\nThey attend a Title I school, which means there is a high enough p
ercentage of free and reduced-price lunch to qualify. Our school is an \"ope
n 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; th
ey are like sponges, absorbing all the information and experiences and keep
on wanting more.With these resources such as the comfy red throw pillows and
the whimsical nautical hanging decor and the blue fish nets, I will be able
to help create the mood in our classroom setting to be one of a themed nauti
cal environment. Creating a classroom environment is very important in the s
uccess in each and every child's education. The nautical photo props will be
used with each child as they step foot into our classroom for the first time
on Meet the Teacher evening. I'll take pictures of each child with them, hav
e them developed, and then hung in our classroom ready for their first day o
f 4th grade. This kind gesture will set the tone before even the first day
of school! The nautical thank you cards will be used throughout the year by
the students as they create thank you cards to their team groups.\r\n\r\nYou
r generous donations will help me to help make our classroom a fun, invitin
g, learning environment from day one.\r\n\r\nIt costs lost of money out of m
y own pocket on resources to get our classroom ready. Please consider helpin
g with this project to make our new school year a very successful one. Thank
you!nannan
==========
My kindergarten students have varied disabilities ranging from speech and la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations. \r\n\r\nThe materials we have are the ones I seek out for my stu
dents. I teach in a Title I school where most of the students receive free o
r reduced price lunch. Despite their disabilities and limitations, my stude
nts love coming to school and come eager to learn and explore.Have you ever
felt like you had ants in your pants and you needed to groove and move as yo
u were in a meeting? This is how my kids feel all the time. The want to be a
ble to move as they learn or so they say.Wobble chairs are the answer and I
love then because they develop their core, which enhances gross motor and in
Turn fine motor skills. \r\nThey also want to learn through games, my kids d
on't want to sit and do worksheets. They want to learn to count by jumping a
nd playing. Physical engagement is the key to our success. The number toss a
nd color and shape mats can make that happen. My students will forget they a
re doing work and just have the fun a 6 year old deserves.nannan
==========

The mediocre teacher tells. The good teacher explains. The superior teacher
demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school
has 803 students which is makeup is 97.6% African-American, making up the la
rgest segment of the student body. A typical school in Dallas is made up of
23.2% African-American students. Most of the students are on free or reduced
lunch. We aren't receiving doctors, lawyers, or engineers children from rich
backgrounds or neighborhoods. As an educator I am inspiring minds of young c
hildren and we focus not only on academics but one smart, effective, efficie
nt, and disciplined students with good character.In our classroom we can uti

```
lize the Bluetooth for swift transitions during class. I use a speaker which
doesn't amplify the sound enough to receive the message. Due to the volume o
f my speaker my students can't hear videos or books clearly and it isn't mak
ing the lessons as meaningful. But with the bluetooth speaker my students wi
ll be able to hear and I can stop, pause and replay it at any time.\r\nThe c
art will allow me to have more room for storage of things that are needed fo
r the day and has an extra part to it I can use.  The table top chart has al
l of the letter, words and pictures for students to learn about different le
tters and it is more accessible.nannan
==================================================
```

In [11]:

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

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

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

In [12]:

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

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

In [13]:

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

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

In [14]:

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

In [15]:

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

In [16]:

```
# 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 = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████| 10924
8/109248 [02:19<00:00, 781.25it/s]
```

In [17]:

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

Out[17]:

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

1.4 Preprocessing of `project_title`

In [18]:

```
# similarly you can preprocess the titles also
project_data['project_title'].head(2)
```

Out[18]:

```
0    Educational Support for English Learners at Home
1               Wanted: Projector for Hungry Learners
Name: project_title, dtype: object
```

In [19]:

```
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[10])
print("="*50)
print(project_data['project_title'].values[100])
print("="*50)
print(project_data['project_title'].values[1500])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
Reading Changes Lives
==================================================
21st Century learners, 21st century technology!
==================================================
Listening Center
==================================================
```

In [20]:

```
preprocessed_titles = []

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

```
100%|██████████| 109248/
109248 [00:06<00:00, 17939.42it/s]
```

In [21]:

```
preprocessed_titles = pd.DataFrame(preprocessed_titles)
project_data['project_title'] = preprocessed_titles
project_data['project_title'].head(10)
```

Out[21]:

```
0            educational support english learners home
1                     wanted projector hungry learners
2      soccer equipment awesome middle school students
3                               techie kindergarteners
4                               interactive math tools
5    flexible seating mrs jarvis terrific third gra...
6        chromebooks special education reading program
7                                     it 21st century
8                          targeting more success class
9                  just for love reading pure pleasure
Name: project_title, dtype: object
```

In [22]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-gr
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index
price_data.head(2)
```

Out[22]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [23]:

```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [24]:

```
project_data['project_title'] = project_data['project_title'].fillna('')
```

In [25]:

```
#randomly sampling 50000 points
X_p = project_data[project_data['project_is_approved']==1].sample(frac = 0.4)
X_n = project_data[project_data['project_is_approved']==0].sample(frac = 1)
project_data = pd.concat([X_p,X_n])
```

In [26]:

```
#shuffling the dataframe
#https://stackoverflow.com/questions/15772009/shuffling-permutating-a-dataframe-in-pandas/3
project_data.reindex(np.random.permutation(project_data.index))
```

Out[26]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submit
56045	7910	p178904	46c4e0ca7bed1164bde4cfdb804fe11b	Ms.	NY	2017-0
30236	171044	p071395	18e9d7c078e7b0340caffb74e044a88e	Mrs.	TX	2016-1
39226	53928	p242736	92d6145bf112b1d2821ece5f50b615ad	Mrs.	SC	2017-0
22204						

1.5 Preparing data for models

In [27]:

```
project_data.columns
```

Out[27]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay', 'price',
       'quantity'],
      dtype='object')
```

we are going to consider

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

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

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

Assignment 8: DT

1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**

   - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
   - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
   - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
   - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

2. **Hyper paramter tuning (best `depth` in range [1, 5, 10, 50, 100, 500, 100], and the best `min_samples_split` in range [5, 10, 100, 500])**

   - 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 or simple cross validation data
   - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Graphviz**

   - Visualize your decision tree with Graphviz. It helps you to understand how a decision is being made, given a new vector.
   - Since feature names are not obtained from word2vec related models, visualize only BOW & TFIDF decision trees using Graphviz
   - Make sure to print the words in each node of the decision tree instead of printing its index.
   - Just for visualization purpose, limit max_depth to 2 or 3 and either embed the generated images of graphviz in your notebook, or directly upload them as .png files.

4. **Representation of results**

   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure

- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.

- Along with plotting ROC curve, you need to print the confusion matrix (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

- Once after you plot the confusion matrix with the test data, get all the `false positive data points`
    - Plot the WordCloud WordCloud (https://www.geeksforgeeks.org/generating-word-cloud-python/)
    - Plot the box plot with the `price` of these `false positive data points`
    - Plot the pdf with the `teacher_number_of_previously_posted_projects` of these `false positive data points`

5. **[Task-2]**
    - Select 5k best features from features of Set 2 using`feature_importances_` (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html), discard all the other remaining features and then apply any of the model of you choice i.e. (Dession tree, Logistic Regression, Linear SVM), you need to do hyperparameter tuning corresponding to the model you selected and procedure in step 2 and step 3

6. **Conclusion**
    - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link (http://zetcode.com/python/prettytable/)

2. Decision Tree

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

In [28]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your cod
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'],axis=1)
X=project_data
```

In [29]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
X_train , X_cv, y_train, y_cv = train_test_split(X_train,y_train,test_size=0.3)
```

In [30]:

```
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)
```

```
(26275, 20) (26275,)
(11261, 20) (11261,)
(16088, 20) (16088,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

Categorical Features

In [31]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your cod
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer1 = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, bi
vectorizer1.fit(X_train['clean_categories'].values)
print(vectorizer1.get_feature_names())

categories_one_hot_train = vectorizer1.transform(X_train['clean_categories'].values)
print("Shape of matrix after one hot encoding ",categories_one_hot_train.shape)
categories_one_hot_test = vectorizer1.transform(X_test['clean_categories'].values)
print("Shape of matrix after one hot encoding ",categories_one_hot_test.shape)
categories_one_hot_cv = vectorizer1.transform(X_cv['clean_categories'].values)
print("Shape of matrix after one hot encoding ",categories_one_hot_cv.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encoding  (26275, 9)
Shape of matrix after one hot encoding  (16088, 9)
Shape of matrix after one hot encoding  (11261, 9)
```

In [32]:

```
vectorizer1.get_feature_names()
```

Out[32]:

```
['Warmth',
 'Care_Hunger',
 'History_Civics',
 'Music_Arts',
 'AppliedLearning',
 'SpecialNeeds',
 'Health_Sports',
 'Math_Science',
 'Literacy_Language']
```

In [33]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer2 = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
vectorizer2.fit(X_train['clean_subcategories'].values)
print(vectorizer2.get_feature_names())


sub_categories_one_hot_train = vectorizer2.transform(X_train['clean_subcategories'].values)
print("Shape of matrix after one hot encoding ",sub_categories_one_hot_train.shape)
sub_categories_one_hot_cv = vectorizer2.transform(X_cv['clean_subcategories'].values)
print("Shape of matrix after one hot encoding ",sub_categories_one_hot_cv.shape)
sub_categories_one_hot_test = vectorizer2.transform(X_test['clean_subcategories'].values)
print("Shape of matrix after one hot encoding ",sub_categories_one_hot_test.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducat
ion', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Characte
rEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_
Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness',
'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encoding  (26275, 30)
Shape of matrix after one hot encoding  (11261, 30)
Shape of matrix after one hot encoding  (16088, 30)
```

In [34]:

```
vectorizer2.get_feature_names()
```

Out[34]:

```
['Economics',
 'CommunityService',
 'FinancialLiteracy',
 'ParentInvolvement',
 'Extracurricular',
 'Civics_Government',
 'ForeignLanguages',
 'NutritionEducation',
 'Warmth',
 'Care_Hunger',
 'SocialSciences',
 'PerformingArts',
 'CharacterEducation',
 'TeamSports',
 'Other',
 'College_CareerPrep',
 'Music',
 'History_Geography',
 'Health_LifeScience',
 'EarlyDevelopment',
 'ESL',
 'Gym_Fitness',
 'EnvironmentalScience',
 'VisualArts',
 'Health_Wellness',
 'AppliedSciences',
 'SpecialNeeds',
 'Literature_Writing',
 'Mathematics',
 'Literacy']
```

In [35]:

```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_categ
# https://machinelearningmastery.com/how-to-one-hot-encode-sequence-data-in-python/
vectorizer3 = CountVectorizer()
vectorizer3.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer3.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer3.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer3.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer3.get_feature_names())
```

```
After vectorizations
(26275, 51) (26275,)
(11261, 51) (11261,)
(16088, 51) (16088,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'i
a', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo',
'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'o
r', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv',
'wy']
```

In [36]:

```
vectorizer3.get_feature_names()
```

Out[36]:

```
['ak',
 'al',
 'ar',
 'az',
 'ca',
 'co',
 'ct',
 'dc',
 'de',
 'fl',
 'ga',
 'hi',
 'ia',
 'id',
 'il',
 'in',
 'ks',
 'ky',
 'la',
 'ma',
 'md',
 'me',
 'mi',
 'mn',
 'mo',
 'ms',
 'mt',
 'nc',
 'nd',
 'ne',
 'nh',
 'nj',
 'nm',
 'nv',
 'ny',
 'oh',
 'ok',
 'or',
 'pa',
 'ri',
 'sc',
 'sd',
 'tn',
 'tx',
 'ut',
 'va',
 'vt',
 'wa',
 'wi',
 'wv',
 'wy']
```

In [37]:

```
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna('null')
```

In [38]:

```
# Please do the similar feature encoding with state, teacher_prefix and project_grade_categ
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np
vectorizer4 = CountVectorizer()
vectorizer4.fit(X_train['teacher_prefix'].values.astype('U')) # fit has to happen only on t

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_onehotencode = vectorizer4.transform(X_train['teacher_prefix'].values.astyp
X_cv_teacher_onehotencode = vectorizer4.transform(X_cv['teacher_prefix'].values.astype('U')
X_test_teacher_onehotencode = vectorizer4.transform(X_test['teacher_prefix'].values.astype(

print("After vectorizations")
print(X_train_teacher_onehotencode.shape, y_train.shape)
print(X_cv_teacher_onehotencode.shape, y_cv.shape)
print(X_test_teacher_onehotencode.shape, y_test.shape)
```

```
After vectorizations
(26275, 5) (26275,)
(11261, 5) (11261,)
(16088, 5) (16088,)
```

In [39]:

```
vectorizer4.get_feature_names()
```

Out[39]:

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
```

In [40]:

```
#This step is to intialize a vectorizer with vocab from train data
from collections import Counter
my_counter4 = Counter()
for word in X_train['project_grade_category'].values:
    my_counter4.update(word.split())

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_grade_category_dict = dict(my_counter4)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items(), key=l
```

In [41]:

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

values = (np.array(X_test['project_grade_category']))
print(values[1:10])

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values.astype(str))
print(integer_encoded[1:10])

# binary encode

project_grade_category_onehot_encoder_test = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
project_grade_category_onehot_encoder_test = project_grade_category_onehot_encoder_test.fit


print("Shape of matrix after one hot encoding ",project_grade_category_onehot_encoder_test.
```

```
['Grades 3-5' 'Grades 3-5' 'Grades 6-8' 'Grades 3-5' 'Grades PreK-2'
 'Grades 9-12' 'Grades 6-8' 'Grades PreK-2' 'Grades 3-5']
[0 0 1 0 3 2 1 3 0]
Shape of matrix after one hot encoding  (16088, 4)
```

In [42]:

```
# binary encode

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

values = (np.array(X_train['project_grade_category']))
print(values[1:10])

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values.astype(str))
print(integer_encoded[1:10])

# binary encode

project_grade_category_onehot_encoder_train = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
project_grade_category_onehot_encoder_train = project_grade_category_onehot_encoder_train.f


print("Shape of matrix after one hot encoding ",project_grade_category_onehot_encoder_trair
```

```
['Grades 6-8' 'Grades 6-8' 'Grades PreK-2' 'Grades PreK-2' 'Grades 9-12'
 'Grades PreK-2' 'Grades PreK-2' 'Grades 3-5' 'Grades 6-8']
[1 1 3 3 2 3 3 0 1]
Shape of matrix after one hot encoding  (26275, 4)
```

In [43]:

```
# binary encode

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder

values = (np.array(X_cv['project_grade_category']))
print(values[1:10])

label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(values.astype(str))
print(integer_encoded[1:10])

# binary encode

project_grade_category_onehot_encoder_cv = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(len(integer_encoded), 1)
project_grade_category_onehot_encoder_cv = project_grade_category_onehot_encoder_cv.fit_tra


print("Shape of matrix after one hot encoding ",project_grade_category_onehot_encoder_cv.sh
```

```
['Grades 6-8' 'Grades 9-12' 'Grades 6-8' 'Grades 9-12' 'Grades 3-5'
 'Grades PreK-2' 'Grades PreK-2' 'Grades 6-8' 'Grades PreK-2']
[1 2 1 2 0 3 3 1 3]
Shape of matrix after one hot encoding  (11261, 4)
```

In [44]:

```
vectorizer5 = ['Grades PreK-2','Grades 3-5','Grades 9-12','Grades 6-8']
```

Tfidf

In [45]:

```
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-np
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['essay'].values)
text_tfidf_train = vectorizer.transform(X_train['essay'])
print("Shape of matrix after one hot encodig ",text_tfidf_train.shape)
text_tfidf_test = vectorizer.transform(X_test['essay'])
print("Shape of matrix after one hot encodig ",text_tfidf_test.shape)
text_tfidf_cv = vectorizer.transform(X_cv['essay'])
print("Shape of matrix after one hot encodig ",text_tfidf_cv.shape)
```

```
Shape of matrix after one hot encodig  (26275, 9752)
Shape of matrix after one hot encodig  (16088, 9752)
Shape of matrix after one hot encodig  (11261, 9752)
```

In [46]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(X_train['project_title'].values.astype('U'))
title_tfidf_train = vectorizer.transform(X_train['project_title'].values.astype('U'))
print("Shape of matrix after one hot encodig ",title_tfidf_train.shape)
title_tfidf_test = vectorizer.transform(X_test['project_title'].values.astype('U'))
print("Shape of matrix after one hot encodig ",title_tfidf_test.shape)
title_tfidf_cv = vectorizer.transform(X_cv['project_title'].values.astype('U'))
print("Shape of matrix after one hot encodig ",title_tfidf_cv.shape)
```

```
Shape of matrix after one hot encodig  (26275, 1373)
Shape of matrix after one hot encodig  (16088, 1373)
Shape of matrix after one hot encodig  (11261, 1373)
```

BOW

In [47]:

```
# please write all the code with proper documentation, and proper titles for each subsectio
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your cod
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
from sklearn.feature_extraction.text import CountVectorizer
vectorizer6 = CountVectorizer(min_df=10)
vectorizer6.fit(X_train['essay'].values)

text_bow_train = vectorizer6.transform(X_train['essay'].values)
text_bow_cv = vectorizer6.transform(X_cv['essay'].values)
text_bow_test = vectorizer6.transform(X_test['essay'].values)

print(text_bow_train.shape, y_train.shape)
print(text_bow_cv.shape, y_cv.shape)
print(text_bow_test.shape, y_test.shape)
```

```
(26275, 9752) (26275,)
(11261, 9752) (11261,)
(16088, 9752) (16088,)
```

In [48]:

```
vectorizer7 = CountVectorizer(min_df=10)
vectorizer7.fit(X_train['project_title'].values.astype('U'))

title_bow_train = vectorizer7.transform(X_train['project_title'].values.astype('U'))
title_bow_cv = vectorizer7.transform(X_cv['project_title'].values.astype('U'))
title_bow_test = vectorizer7.transform(X_test['project_title'].values.astype('U'))

print(title_bow_train.shape, y_train.shape)
print(title_bow_cv.shape, y_cv.shape)
print(title_bow_test.shape, y_test.shape)
```

```
(26275, 1373) (26275,)
(11261, 1373) (11261,)
(16088, 1373) (16088,)
```

AVG W2V

In [49]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# ============================
Output:

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

# ============================

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

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickl

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


'''
```

Out[49]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4
```

```
084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGloveMod
el(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFil
e,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n
splitLine = line.split()\n        word = splitLine[0]\n        embedding = n
p.array([float(val) for val in splitLine[1:]])\n        model[word] = embedd
ing\n    print ("Done.",len(model)," words loaded!")\n    return model\nmode
l = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ==========================
\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\nDone.
 1917495  words loaded!\n\n# ==========================\n\nwords = []\nfor
i in preproced_texts:\n    words.extend(i.split(\' \'))\n\nfor i in preproce
d_titles:\n    words.extend(i.split(\' \'))\nprint("all the words in the cou
pus", len(words))\nwords = set(words)\nprint("the unique words in the coupu
s", len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprin
t("The number of words that are present in both glove vectors and our coupu
s",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,
3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in wo
rds:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint
("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pi
ckle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-
load-variables-in-python/\n\nimport (http://www.jessicayung.com/how-to-use-p
ickle-to-save-and-load-variables-in-python/\n\nimport) pickle\nwith open(\'g
love_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [50]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickl
# 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 [51]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay']): # 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_train.append(vector)

print(len(avg_w2v_vectors_train))
print(len(avg_w2v_vectors_train[0]))


avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay']): # 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_test.append(vector)

print(len(avg_w2v_vectors_test))
print(len(avg_w2v_vectors_test[0]))


avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay']): # 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_cv.append(vector)

print(len(avg_w2v_vectors_cv))
print(len(avg_w2v_vectors_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████| 262
75/26275 [00:26<00:00, 992.77it/s]

26275
300

100%|████████████████████████████████████████████████████████████| 1608
8/16088 [00:12<00:00, 1331.21it/s]

16088
300
```

```
100%|████████████████████████████████████████████████████████████| 1126
1/11261 [00:08<00:00, 1339.82it/s]

11261
300
```

In [52]:

```
avg_w2v_vectors_train =  np.array(avg_w2v_vectors_train)
avg_w2v_vectors_test  =  np.array(avg_w2v_vectors_test)
avg_w2v_vectors_cv    =  np.array(avg_w2v_vectors_cv)
```

In [53]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_train_title = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(X_train['project_title'].values.astype('U')): # for each review/senter
    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_train_title.append(vector)

print(len(avg_w2v_vectors_train_title))
print(len(avg_w2v_vectors_train_title[0]))

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_test_title = []; # the avg-w2v for each sentence/review is stored in this l
for sentence in tqdm(X_test['project_title'].values.astype('U')): # for each review/sentenc
    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_test_title.append(vector)

print(len(avg_w2v_vectors_test_title))
print(len(avg_w2v_vectors_test_title[0]))

# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_cv_title = []; # the avg-w2v for each sentence/review is stored in this lis
for sentence in tqdm(X_cv['project_title'].values.astype('U')): # 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_cv_title.append(vector)

print(len(avg_w2v_vectors_cv_title))
print(len(avg_w2v_vectors_cv_title[0]))
```

```
100%|████████████████████████████████████████████████████████████| 2627
5/26275 [00:00<00:00, 31982.87it/s]

26275
300

100%|████████████████████████████████████████████████████████████| 1608
8/16088 [00:00<00:00, 23393.04it/s]
```

```
16088
300

100%|██████████████████████████████████████████████████████████████████| 1126
1/11261 [00:00<00:00, 30207.62it/s]

11261
300
```

In [54]:

```
avg_w2v_vectors_train_title =  np.array(avg_w2v_vectors_train_title)
avg_w2v_vectors_test_title  =  np.array(avg_w2v_vectors_test_title)
avg_w2v_vectors_cv_title    =  np.array(avg_w2v_vectors_cv_title)
```

TF-IDF AVG W2V

In [55]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [56]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            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_train.append(vector)

print(len(tfidf_w2v_vectors_train))
print(len(tfidf_w2v_vectors_train[0]))


# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            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_test.append(vector)

print(len(tfidf_w2v_vectors_test))
print(len(tfidf_w2v_vectors_test[0]))


# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            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_cv.append(vector)
```

```
print(len(tfidf_w2v_vectors_cv))
print(len(tfidf_w2v_vectors_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████| 26
275/26275 [04:43<00:00, 92.73it/s]

26275
300

100%|████████████████████████████████████████████████████████████| 16
088/16088 [02:58<00:00, 90.21it/s]

16088
300

100%|████████████████████████████████████████████████████████████| 112
61/11261 [01:42<00:00, 109.38it/s]

11261
300
```

In [57]:

```
tfidf_w2v_vectors_train = np.array(tfidf_w2v_vectors_train)
tfidf_w2v_vectors_test  = np.array(tfidf_w2v_vectors_test)
tfidf_w2v_vectors_cv    = np.array(tfidf_w2v_vectors_cv)
```

In [58]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['project_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [59]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_title_train = []; # the avg-w2v for each sentence/review is stored in thi
for sentence in tqdm(X_train['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_train.append(vector)

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


# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_title_test = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(X_test['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_test.append(vector)

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


# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_title_cv = []; # the avg-w2v for each sentence/review is stored in this l
for sentence in tqdm(X_cv['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_title_cv.append(vector)
```

```
print(len(tfidf_w2v_vectors_title_cv))
```

```
100%|████████████████████████████████████████████████████████████| 2627
5/26275 [00:01<00:00, 17655.97it/s]

26275
300

100%|████████████████████████████████████████████████████████████| 1608
8/16088 [00:00<00:00, 18250.81it/s]

16088
300

100%|████████████████████████████████████████████████████████████| 1126
1/11261 [00:00<00:00, 17687.82it/s]

11261
```

In [60]:

```
tfidf_w2v_vectors_title_train = np.array(tfidf_w2v_vectors_title_train)
tfidf_w2v_vectors_title_test  = np.array(tfidf_w2v_vectors_title_test)
tfidf_w2v_vectors_title_cv    = np.array(tfidf_w2v_vectors_title_cv)
```

In [61]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.prepro
from sklearn.preprocessing import StandardScaler
# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.... 399.
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standard dev

# Now standardize the data with above maen and variance.
price_standardized_train = price_scalar.transform(X_train['price'].values.reshape(-1, 1))

price_scalar.fit(X_test['price'].values.reshape(-1,1))
price_standardized_test  = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
price_scalar.fit(X_cv['price'].values.reshape(-1,1))
price_standardized_cv    = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
```

In [62]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using # array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1)
prev_projects_train = normalizer.transform(X_train['teacher_number_of_previously_posted_prc
prev_projects_cv = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'
prev_projects_test = normalizer.transform(X_test['teacher_number_of_previously_posted_proje
print("After vectorizations")
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
```

```
After vectorizations
(26275, 1) (26275,)
(11261, 1) (11261,)
(16088, 1) (16088,)
```

In [63]:

```
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead: # array=[105.22 215.96
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature # array.reshape(1, -1)  if it cont
normalizer.fit(X_train['quantity'].values.reshape(-1,1))
quantity_train = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
quantity_cv = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
quantity_test = normalizer.transform(X_test['quantity'].values.reshape(-1,1))
print("After vectorizations")
print(quantity_train.shape, y_train.shape)
print(quantity_cv.shape, y_cv.shape)
print(quantity_test.shape, y_test.shape)
```

```
After vectorizations
(26275, 1) (26275,)
(11261, 1) (11261,)
(16088, 1) (16088,)
```

2.4 Appling Decision Tree on different kind of featurization as mentioned in the instructions

Apply Decision Tree 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

In [64]:

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

2.4.1 Applying Decision Trees on BOW, SET 1

In [65]:

```
# Please write all the code with proper documentation
# Please write all the code with proper documentation
from scipy.sparse import hstack
X1 = hstack((categories_one_hot_train, sub_categories_one_hot_train, X_train_state_ohe, X_t
X2 = hstack((categories_one_hot_test, sub_categories_one_hot_test, X_test_state_ohe, X_test
X3 = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, X_cv_state_ohe, X_cv_teacher
print("Final Data matrix")
print(X1.shape, y_train.shape)
print(X3.shape, y_cv.shape)
print(X2.shape, y_test.shape)
```

```
Final Data matrix
(26275, 11227) (26275,)
(11261, 11227) (11261,)
(16088, 11227) (16088,)
```

2.4.1.1 Graphviz visualization of Decision Tree on BOW, SET 1

In [66]:

```
# Please write all the code with proper documentation
all_features=[]
all_features.extend(vectorizer1.get_feature_names())
all_features.extend(vectorizer2.get_feature_names())
all_features.extend(vectorizer3.get_feature_names())
all_features.extend(vectorizer4.get_feature_names())
all_features.extend(vectorizer5)
all_features.extend(vectorizer6.get_feature_names())
all_features.extend(vectorizer7.get_feature_names())
all_features.append('price')
all_features.append('teacher_number_of_previously_posted_projects')
all_features.append('quantity')
```

In [67]:

```
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
```

In [68]:

```
d_tree = DecisionTreeClassifier(max_depth = 3,class_weight = 'balanced')
clf = d_tree.fit(X1,y_train)
```

In [69]:

```
import os
os.environ["PATH"] += os.pathsep + 'C:/Program Files (x86)/Graphviz2.38/bin/'
```

In [71]:

```
# Visualize data import graphviz
import graphviz
from sklearn import tree
from graphviz import Source
dot_data = tree.export_graphviz(d_tree, out_file=None, feature_names=all_features)
graph = graphviz.Source(dot_data)
graph.render("Bow tree",view = True)
```

Out[71]:

```
'Bow tree.pdf'
```

In [75]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.ht
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

d_tree = DecisionTreeClassifier(class_weight='balanced')
parameters = {'max_depth':[5, 10, 50, 100, 200, 500, 1000], 'min_samples_split': [5, 10,20,
clf = GridSearchCV(d_tree, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X1, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

```
Wall time: 49min 28s
```

In [76]:

```
train_auc
```

Out[76]:

```
array([ 0.63742556,  0.63723294,  0.6369854 ,  0.63658015,  0.63650291,
        0.63623634,  0.7521476 ,  0.74890503,  0.74214279,  0.73728132,
        0.73094094,  0.72567512,  0.99081393,  0.98309921,  0.96859213,
        0.95902397,  0.93508713,  0.91085634,  0.99902856,  0.99520867,
        0.986318  ,  0.97512956,  0.95666527,  0.93573105,  0.99983172,
        0.99688351,  0.98758138,  0.97796583,  0.95966755,  0.9366037 ,
        0.9998362 ,  0.99712267,  0.98778117,  0.97890931,  0.96009353,
        0.93703391,  0.99983409,  0.99712322,  0.98772879,  0.97776184,
        0.9608254 ,  0.93799766])
```

plot auc vs hyper-parameter plots as seaborn heatmaps, with rows as min sample split , columns as max_depth, and values inside the cell representing AUC Score. By looking train and test heatmaps, you will get

best min sample split , and depth for both train and test data

Plotting Train-Auc heatmap

In [77]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['train_auc'] = train_auc
df.head(5)
```

Out[77]:

	max_depth	min_split	train_auc
0	5	5	0.637426
1	10	10	0.637233
2	50	20	0.636985
3	100	30	0.636580
4	200	50	0.636503

In [78]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[78]:

		train_auc
max_depth	**min_split**	
5	**5**	0.637426
	10	0.748905
	20	0.968592
	30	0.975130
	50	0.959668
	80	0.937034
10	**5**	0.999834
	10	0.637233
	20	0.742143
	30	0.959024

In [79]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[79]:

	max_depth	min_split	train_auc
0	5	5	0.637426
1	5	10	0.748905
2	5	20	0.968592
3	5	30	0.975130
4	5	50	0.959668

In [80]:

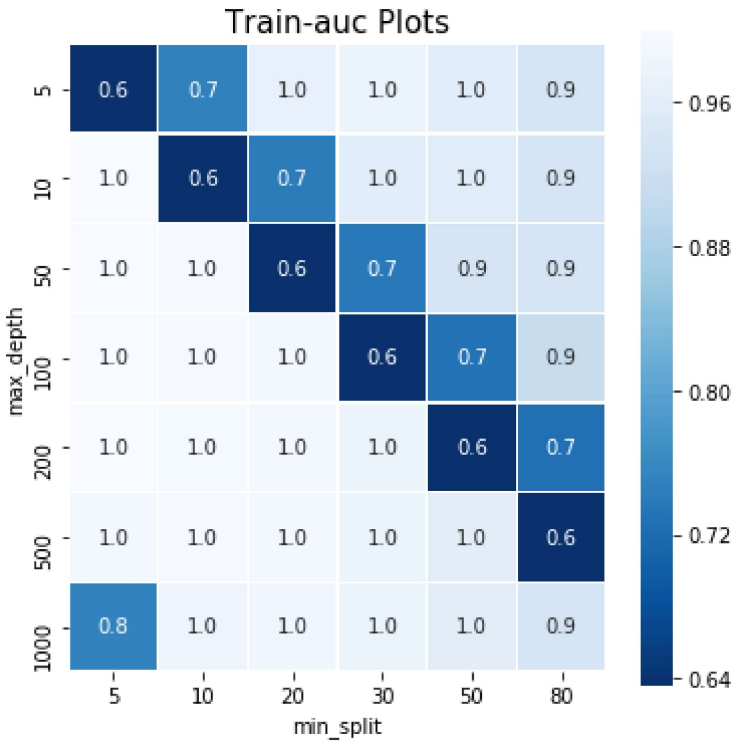
```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','train_auc').head()
```

Out[80]:

min_split	5	10	20	30	50	80
max_depth						
5	0.637426	0.748905	0.968592	0.975130	0.959668	0.937034
10	0.999834	0.637233	0.742143	0.959024	0.956665	0.936604
50	0.999836	0.997123	0.636985	0.737281	0.935087	0.935731
100	0.999832	0.997123	0.987729	0.636580	0.730941	0.910856
200	0.999029	0.996884	0.987781	0.977762	0.636503	0.725675

In [81]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','train_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Train-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```



Plotting CV- AUC heatmap

In [84]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['cv_auc'] = cv_auc
df.head(5)
```

Out[84]:

	max_depth	min_split	cv_auc
0	5	5	0.606989
1	10	10	0.606780
2	50	20	0.606639
3	100	30	0.606356
4	200	50	0.606314

In [85]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[85]:

		cv_auc
max_depth	**min_split**	
5	**5**	0.606989
	10	0.611591
	20	0.580598
	30	0.581677
	50	0.587694
	80	0.590037
10	**5**	0.567785
	10	0.606780
	20	0.609005
	30	0.582766

In [86]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[86]:

	max_depth	min_split	cv_auc
0	5	5	0.606989
1	5	10	0.611591
2	5	20	0.580598
3	5	30	0.581677
4	5	50	0.587694

In [87]:

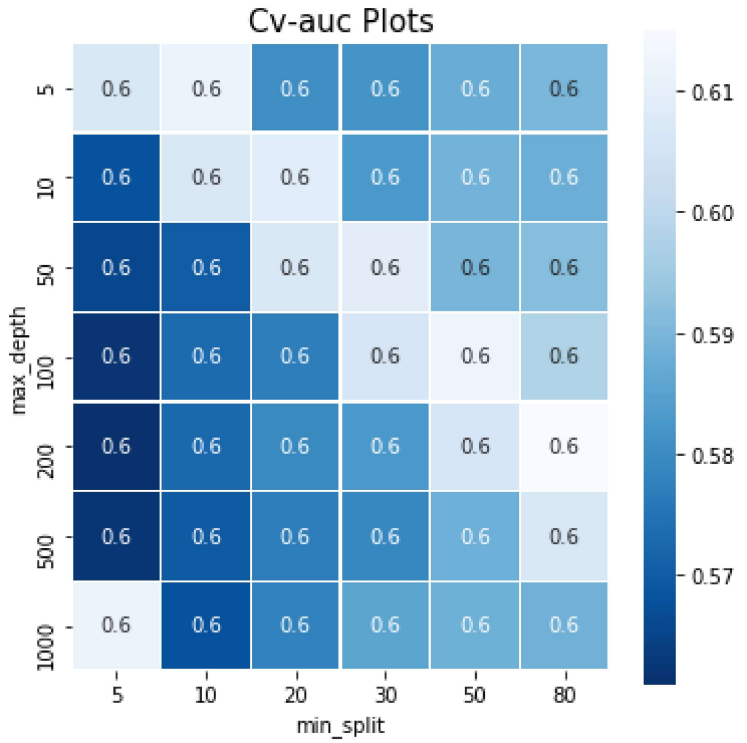
```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','cv_auc').head()
```

Out[87]:

min_split	5	10	20	30	50	80
max_depth						
5	0.606989	0.611591	0.580598	0.581677	0.587694	0.590037
10	0.567785	0.606780	0.609005	0.582766	0.588971	0.588225
50	0.565445	0.570060	0.606639	0.609769	0.589553	0.591785
100	0.561958	0.572987	0.577139	0.606356	0.612669	0.597838
200	0.561005	0.572766	0.579584	0.582831	0.606314	0.615050

In [88]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','cv_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Cv-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```



Optimal Parameters

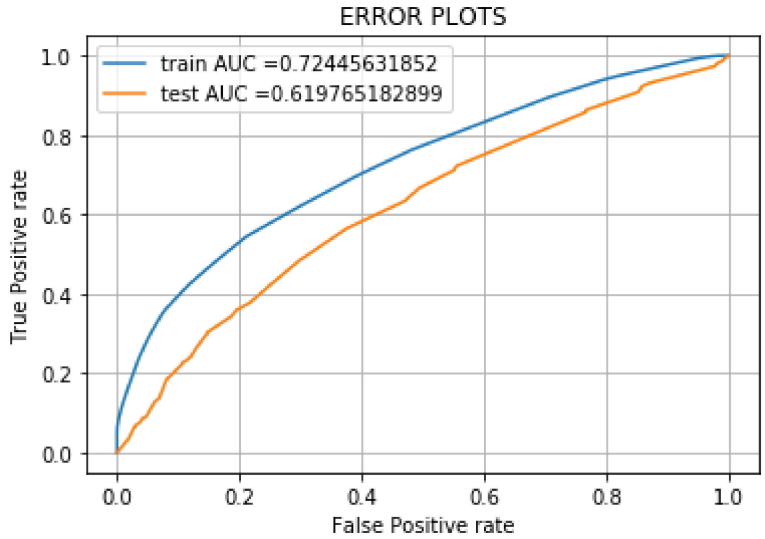
min_sample_split = 30
max_depth = 10

In [90]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 4900
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [91]:

```
from sklearn.metrics import roc_curve, auc


clf= DecisionTreeClassifier( min_samples_split = 30 , max_depth = 10 , class_weight = 'bala
clf.fit(X1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(clf, X1)
y_test_pred = batch_predict(clf, X2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

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("False Positive rate")
plt.ylabel("True Positive rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [92]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

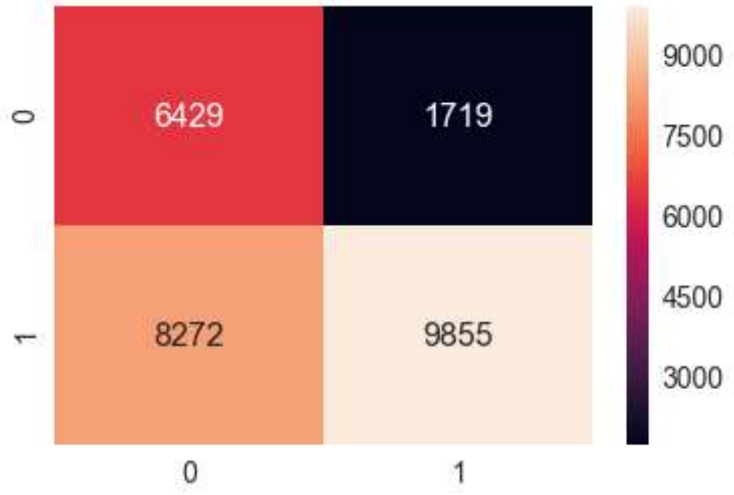
In [93]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
#https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

y_pred_new = clf.predict(X1)

print("Confusion Matrix of train set:\n [ [TN  FP]\n [FN TP] ]\n")
df_cm = pd.DataFrame(confusion_matrix(y_train, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================
========================
Confusion Matrix of train set:
 [ [TN  FP]
 [FN TP] ]
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x209e6519940>
```



In [94]:

```
y_pred_new = clf.predict(X2)

df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x209f4423128>
```

2.4.2 Applying Decision Trees on TFIDF, SET 2

In [95]:

```
# Please write all the code with proper documentation
from scipy.sparse import hstack
X1 = hstack((categories_one_hot_train, sub_categories_one_hot_train, X_train_state_ohe, X_t
X2 = hstack((categories_one_hot_test, sub_categories_one_hot_test, X_test_state_ohe, X_test
X3 = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, X_cv_state_ohe, X_cv_teacher
print("Final Data matrix")
print(X1.shape, y_train.shape)
print(X3.shape, y_cv.shape)
print(X2.shape, y_test.shape)
```

```
Final Data matrix
(26275, 11227) (26275,)
(11261, 11227) (11261,)
(16088, 11227) (16088,)
```

In [99]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.ht
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

d_tree = DecisionTreeClassifier(class_weight='balanced')
parameters = {'max_depth':[5, 10, 50, 100, 200, 500, 1000], 'min_samples_split': [5, 10,20,
clf = GridSearchCV(d_tree, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X1, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

```
Wall time: 59min 35s
```

In [100]:

```
train_auc
```

Out[100]:

```
array([ 0.6501104 ,  0.64997357,  0.64978187,  0.64967506,  0.64917829,
        0.64829928,  0.7636419 ,  0.76120519,  0.75356846,  0.74888549,
        0.74230785,  0.73505128,  0.99729527,  0.99410998,  0.9830581 ,
        0.97214336,  0.9599723 ,  0.93813049,  0.99982847,  0.99797104,
        0.99014244,  0.98186984,  0.96873579,  0.95028095,  0.999903  ,
        0.99809284,  0.99072157,  0.98217783,  0.96925054,  0.94927213,
        0.99990074,  0.99813536,  0.99022173,  0.98279591,  0.9681428 ,
        0.95035067,  0.99989911,  0.99816573,  0.99025806,  0.9827873 ,
        0.96884563,  0.94913626])
```

In [101]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['train_auc'] = train_auc
df.head(5)
```

Out[101]:

	max_depth	min_split	train_auc
0	5	5	0.650110
1	10	10	0.649974
2	50	20	0.649782
3	100	30	0.649675
4	200	50	0.649178

In [102]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[102]:

		train_auc
max_depth	**min_split**	
5	**5**	0.650110
	10	0.761205
	20	0.983058
	30	0.981870
	50	0.969251
	80	0.950351
10	**5**	0.999899
	10	0.649974
	20	0.753568
	30	0.972143

In [103]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[103]:

	max_depth	min_split	train_auc
0	5	5	0.650110
1	5	10	0.761205
2	5	20	0.983058
3	5	30	0.981870
4	5	50	0.969251

In [104]:

```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','train_auc').head()
```

Out[104]:

min_split	5	10	20	30	50	80
max_depth						
5	0.650110	0.761205	0.983058	0.981870	0.969251	0.950351
10	0.999899	0.649974	0.753568	0.972143	0.968736	0.949272
50	0.999901	0.998166	0.649782	0.748885	0.959972	0.950281
100	0.999903	0.998135	0.990258	0.649675	0.742308	0.938130
200	0.999828	0.998093	0.990222	0.982787	0.649178	0.735051

In [105]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','train_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Train-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```

In [106]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['cv_auc'] = cv_auc
df.head(5)
```

Out[106]:

	max_depth	min_split	cv_auc
0	5	5	0.618906
1	10	10	0.618800
2	50	20	0.618911
3	100	30	0.618887
4	200	50	0.618946

In [107]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[107]:

		cv_auc
max_depth	**min_split**	
5	**5**	0.618906
	10	0.618834
	20	0.567585
	30	0.571989
	50	0.571948
	80	0.577103
10	**5**	0.551299
	10	0.618800
	20	0.617765
	30	0.574263

In [108]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[108]:

	max_depth	min_split	cv_auc
0	5	5	0.618906
1	5	10	0.618834
2	5	20	0.567585
3	5	30	0.571989
4	5	50	0.571948

In [109]:

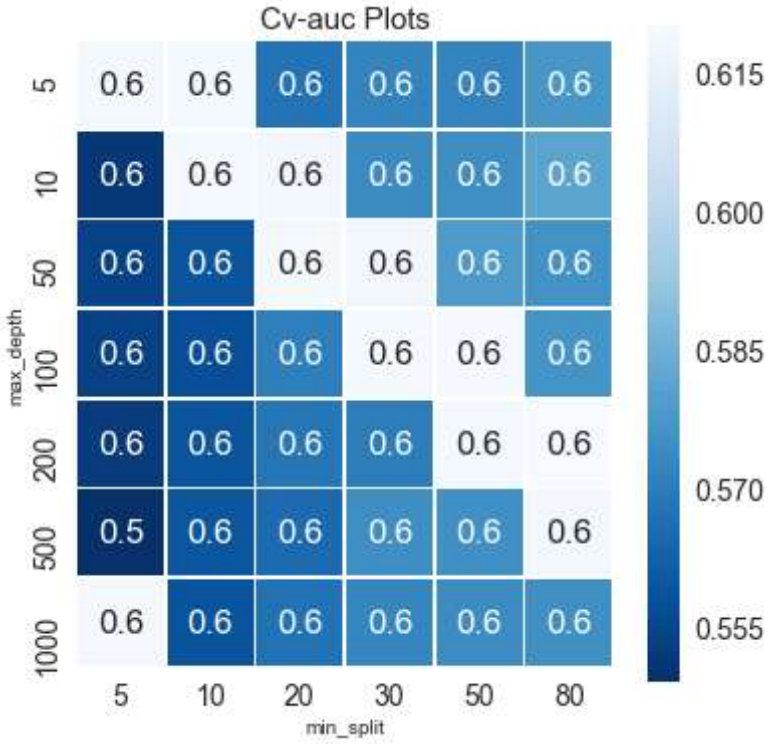
```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','cv_auc').head()
```

Out[109]:

min_split	5	10	20	30	50	80
max_depth						
5	0.618906	0.618834	0.567585	0.571989	0.571948	0.577103
10	0.551299	0.618800	0.617765	0.574263	0.574796	0.581668
50	0.554324	0.559333	0.618911	0.617739	0.578663	0.575978
100	0.553889	0.557298	0.570450	0.618887	0.619732	0.576295
200	0.552542	0.559301	0.568741	0.569963	0.618946	0.620075

In [110]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','cv_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Cv-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```

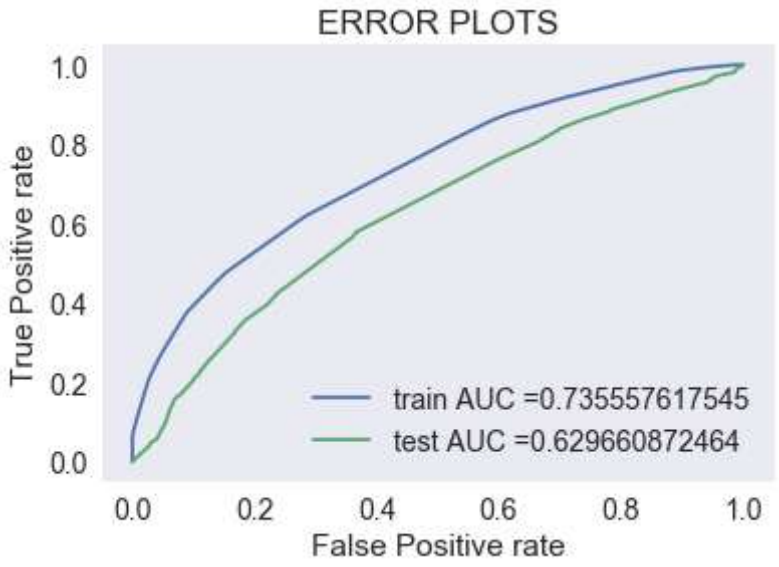


In [111]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 49041%1000 = 4900
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

    return y_data_pred
```

In [112]:

```
from sklearn.metrics import roc_curve, auc


clf= DecisionTreeClassifier( min_samples_split = 30 , max_depth = 10 , class_weight = 'bala
clf.fit(X1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(clf, X1)
y_test_pred = batch_predict(clf, X2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

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("False Positive rate")
plt.ylabel("True Positive rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [113]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [114]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
#https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

y_pred_new = clf.predict(X1)

print("Confusion Matrix of train set:\n [ [TN  FP]\n [FN TP] ]\n")
df_cm = pd.DataFrame(confusion_matrix(y_train, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================
========================
Confusion Matrix of train set:
 [ [TN  FP]
 [FN TP] ]
```

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x209a04c75c0>
```

In [115]:

```
y_pred_new = clf.predict(X2)

df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[115]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2099f9ca400>
```

2.4.2.1 Graphviz visualization of Decision Tree on TFIDF, SET 2

In [116]:

```
# Please write all the code with proper documentation
# Please write all the code with proper documentation
all_features=[]
all_features.extend(vectorizer1.get_feature_names())
all_features.extend(vectorizer2.get_feature_names())
all_features.extend(vectorizer3.get_feature_names())
all_features.extend(vectorizer4.get_feature_names())
all_features.extend(vectorizer5)
all_features.extend(vectorizer6.get_feature_names())
all_features.extend(vectorizer.get_feature_names())
all_features.append('price')
all_features.append('teacher_number_of_previously_posted_projects')
all_features.append('quantity')
```

In [117]:

```
d_tree = DecisionTreeClassifier(max_depth = 3,class_weight='balanced')
clf = d_tree.fit(X1,y_train)
```

In [118]:

```
# Visualize data import graphviz
import graphviz
from sklearn import tree
from graphviz import Source
dot_data = tree.export_graphviz(d_tree, out_file=None, feature_names=all_features)
graph = graphviz.Source(dot_data)
graph.render("Bow tree2",view = True)
```

Out[118]:

```
'Bow tree2.pdf'
```

2.4.3 Applying Decision Trees on AVG W2V, SET 3

In [119]:

```
# Please write all the code with proper documentation
from scipy.sparse import hstack
X1 = hstack((categories_one_hot_train, sub_categories_one_hot_train, X_train_state_ohe, X_t
X2 = hstack((categories_one_hot_test, sub_categories_one_hot_test, X_test_state_ohe, X_test
X3 = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, X_cv_state_ohe, X_cv_teacher
print("Final Data matrix")
print(X1.shape, y_train.shape)
print(X3.shape, y_cv.shape)
print(X2.shape, y_test.shape)
```

```
Final Data matrix
(26275, 11227) (26275,)
(11261, 11227) (11261,)
(16088, 11227) (16088,)
```

In [120]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.ht
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

d_tree = DecisionTreeClassifier(class_weight='balanced')
parameters = {'max_depth':[5, 10, 50, 100, 200, 500, 1000], 'min_samples_split': [5, 10,20,
clf = GridSearchCV(d_tree, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X1, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

```
Wall time: 1h 22min 16s
```

In [121]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['train_auc'] = train_auc
df.head(5)
```

Out[121]:

	max_depth	min_split	train_auc
0	5	5	0.650110
1	10	10	0.649974
2	50	20	0.649782
3	100	30	0.649675
4	200	50	0.649178

In [122]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[122]:

		train_auc
max_depth	**min_split**	
5	**5**	0.650110
	10	0.761319
	20	0.983391
	30	0.981947
	50	0.968230
	80	0.949980
10	**5**	0.999893
	10	0.649974
	20	0.753204
	30	0.975724

In [123]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[123]:

	max_depth	min_split	train_auc
0	5	5	0.650110
1	5	10	0.761319
2	5	20	0.983391
3	5	30	0.981947
4	5	50	0.968230

In [124]:

```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','train_auc').head()
```

Out[124]:

min_split	5	10	20	30	50	80
max_depth						
5	0.650110	0.761319	0.983391	0.981947	0.968230	0.949980
10	0.999893	0.649974	0.753204	0.975724	0.968847	0.951707
50	0.999899	0.998191	0.649782	0.749184	0.958636	0.948303
100	0.999904	0.998190	0.990117	0.649675	0.742091	0.937984
200	0.999869	0.998132	0.990102	0.982618	0.649178	0.735004

In [125]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','train_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Train-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```

In [126]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['cv_auc'] = cv_auc
df.head(5)
```

Out[126]:

	max_depth	min_split	cv_auc
0	5	5	0.619017
1	10	10	0.619020
2	50	20	0.618911
3	100	30	0.618942
4	200	50	0.619001

In [127]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[127]:

		cv_auc
max_depth	**min_split**	
5	**5**	0.619017
	10	0.617575
	20	0.571080
	30	0.574917
	50	0.574985
	80	0.578254
10	**5**	0.549106
	10	0.619020
	20	0.617871
	30	0.577045

In [128]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[128]:

	max_depth	min_split	cv_auc
0	5	5	0.619017
1	5	10	0.617575
2	5	20	0.571080
3	5	30	0.574917
4	5	50	0.574985

In [129]:

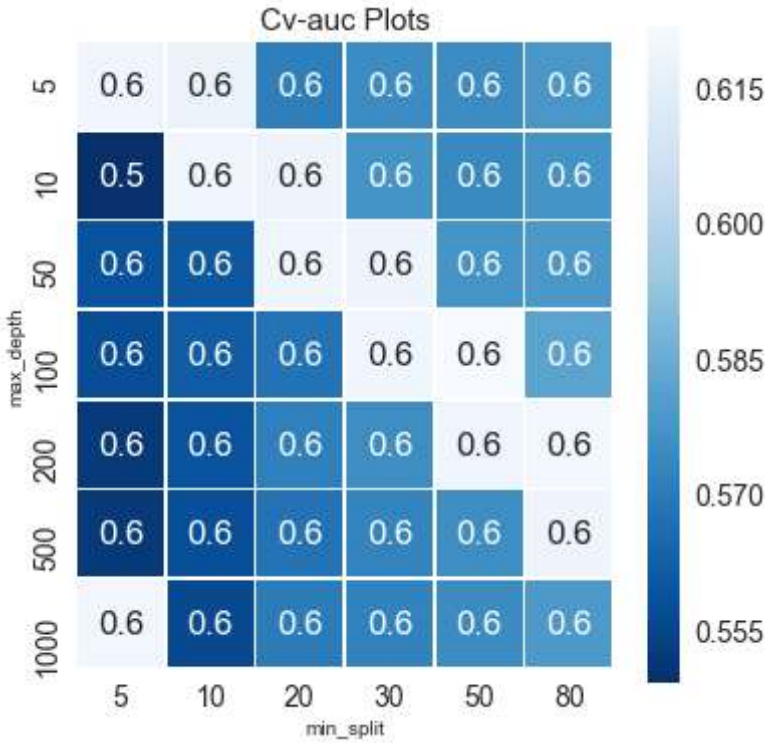
```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','cv_auc').head()
```

Out[129]:

min_split	5	10	20	30	50	80
max_depth						
5	0.619017	0.617575	0.571080	0.574917	0.574985	0.578254
10	0.549106	0.619020	0.617871	0.577045	0.574223	0.576930
50	0.558580	0.560193	0.618911	0.618522	0.577348	0.578528
100	0.557182	0.561532	0.568081	0.618942	0.621715	0.582242
200	0.551718	0.558813	0.571484	0.575295	0.619001	0.620102

In [130]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','cv_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Cv-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```

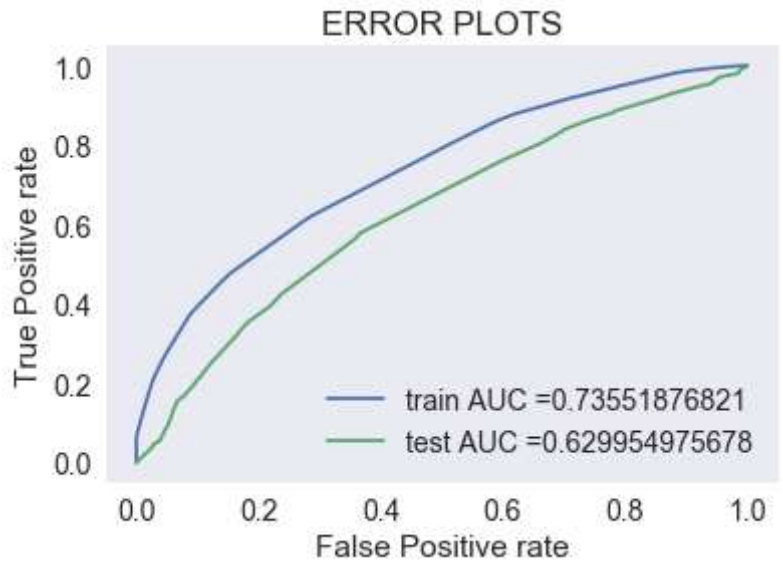


Optimal Parameters

min_sample_split = 50
max_depth = 10

In [131]:

```
from sklearn.metrics import roc_curve, auc


clf= DecisionTreeClassifier( min_samples_split = 30, max_depth = 10, class_weight = 'balanc
clf.fit(X1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(clf, X1)
y_test_pred = batch_predict(clf, X2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

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("False Positive rate")
plt.ylabel("True Positive rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [132]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [133]:

```
print("="*100)
from sklearn.metrics import confusion_matrix

#https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
y_pred_new = clf.predict(X1)

print("Confusion Matrix of train set:\n [ [TN  FP]\n [FN TP] ]\n")
df_cm = pd.DataFrame(confusion_matrix(y_train, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================
========================
Confusion Matrix of train set:
 [ [TN  FP]
 [FN TP] ]
```

Out[133]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x209a048b080>
```

In [134]:

```
y_pred_new = clf.predict(X2)

df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[134]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x209a1a2c320>
```

2.4.4 Applying Decision Trees on TFIDF W2V, SET 4

In [135]:

```
# Please write all the code with proper documentation
from scipy.sparse import hstack
X1 = hstack((categories_one_hot_train, sub_categories_one_hot_train, X_train_state_ohe, X_t
X2 = hstack((categories_one_hot_test, sub_categories_one_hot_test, X_test_state_ohe, X_test
X3 = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, X_cv_state_ohe, X_cv_teacher
print("Final Data matrix")
print(X1.shape, y_train.shape)
print(X3.shape, y_cv.shape)
print(X2.shape, y_test.shape)
```

```
Final Data matrix
(26275, 702) (26275,)
(11261, 702) (11261,)
(16088, 702) (16088,)
```

In [136]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.ht
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

d_tree = DecisionTreeClassifier(class_weight='balanced')
parameters = {'max_depth':[5, 10, 50, 100, 200, 500, 1000], 'min_samples_split': [5, 10,20,
clf = GridSearchCV(d_tree, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(X1, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

```
Wall time: 2h 21min 41s
```

In [137]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['train_auc'] = train_auc
df.head(5)
```

Out[137]:

	max_depth	min_split	train_auc
0	5	5	0.651964
1	10	10	0.651964
2	50	20	0.651964
3	100	30	0.651828
4	200	50	0.651291

In [138]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[138]:

		train_auc
max_depth	**min_split**	
5	**5**	0.651964
	10	0.846550
	20	0.990330
	30	0.976893
	50	0.943073
	80	0.898424
10	**5**	0.999950
	10	0.651964
	20	0.837638
	30	0.976874

In [139]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[139]:

	max_depth	min_split	train_auc
0	5	5	0.651964
1	5	10	0.846550
2	5	20	0.990330
3	5	30	0.976893
4	5	50	0.943073

In [140]:

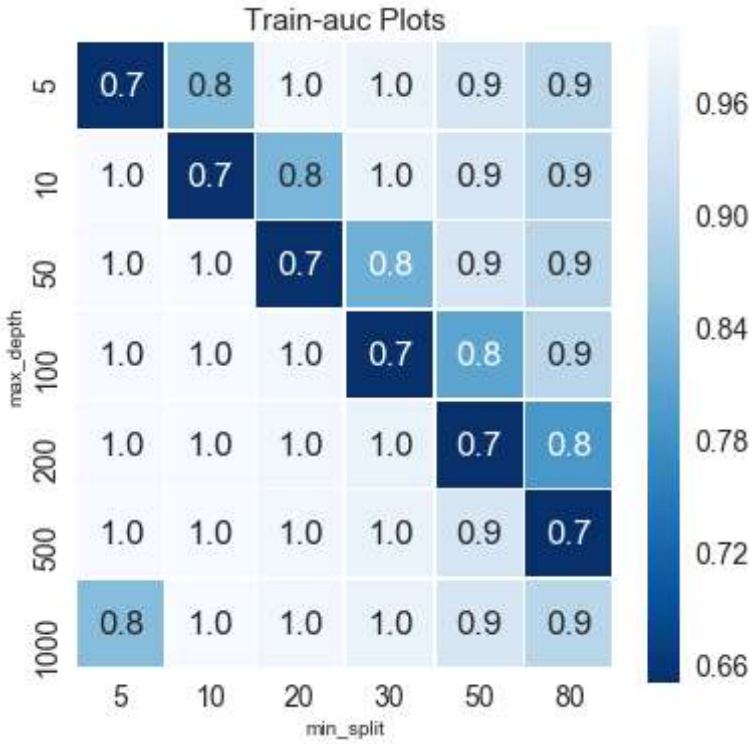
```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','train_auc').head()
```

Out[140]:

min_split	5	10	20	30	50	80
max_depth						
5	0.651964	0.846550	0.990330	0.976893	0.943073	0.898424
10	0.999950	0.651964	0.837638	0.976874	0.943738	0.899184
50	0.999945	0.998757	0.651964	0.828942	0.942740	0.899556
100	0.999949	0.998775	0.990211	0.651828	0.810782	0.899626
200	0.999946	0.998775	0.990248	0.976697	0.651291	0.790990

In [141]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','train_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Train-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```



In [143]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['cv_auc'] = cv_auc
df.head(5)
```

Out[143]:

	max_depth	min_split	cv_auc
0	5	5	0.596731
1	10	10	0.596841
2	50	20	0.596841
3	100	30	0.596826
4	200	50	0.597012

In [144]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[144]:

		cv_auc
max_depth	**min_split**	
5	**5**	0.596731
	10	0.568032
	20	0.536270
	30	0.541055
	50	0.550862
	80	0.554847
10	**5**	0.534236
	10	0.596841
	20	0.569021
	30	0.543020

In [145]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[145]:

	max_depth	min_split	cv_auc
0	5	5	0.596731
1	5	10	0.568032
2	5	20	0.536270
3	5	30	0.541055
4	5	50	0.550862

In [146]:

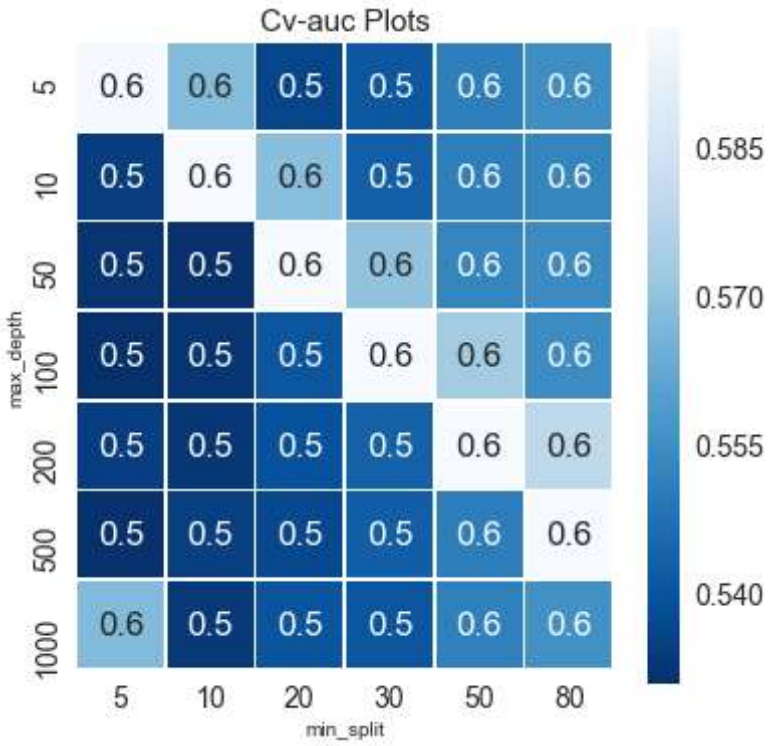
```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','cv_auc').head()
```

Out[146]:

min_split	5	10	20	30	50	80
max_depth						
5	0.596731	0.568032	0.536270	0.541055	0.550862	0.554847
10	0.534236	0.596841	0.569021	0.543020	0.550942	0.552369
50	0.531931	0.531123	0.596841	0.569857	0.552272	0.554278
100	0.530771	0.532209	0.540517	0.596826	0.573034	0.554520
200	0.534461	0.532762	0.539071	0.543566	0.597012	0.578793

In [147]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','cv_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Cv-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```

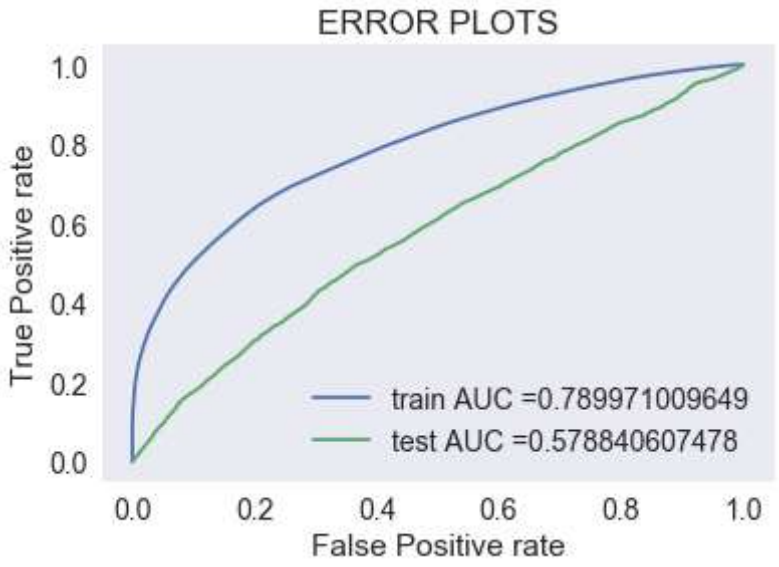


Optimal Parameters

min_sample_split = 50
max_depth = 10

In [148]:

```
from sklearn.metrics import roc_curve, auc


clf= DecisionTreeClassifier( min_samples_split =50, max_depth=10 , class_weight = 'balanced
clf.fit(X1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(clf, X1)
y_test_pred = batch_predict(clf, X2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

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("False Positive rate")
plt.ylabel("True Positive rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [149]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

In [150]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
#https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

y_pred_new = clf.predict(X1)

print("Confusion Matrix of train set:\n [ [TN  FP]\n [FN TP] ]\n")
df_cm = pd.DataFrame(confusion_matrix(y_train, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================
========================
Confusion Matrix of train set:
 [ [TN  FP]
 [FN TP] ]
```

Out[150]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x209a1aac860>
```

In [151]:

```
y_pred_new = clf.predict(X2)

df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[151]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x209a116b3c8>
```

2.5 [Task-2]Getting top 5k features using `feature_importances_`

In [152]:

```
# Please write all the code with proper documentation
from scipy.sparse import hstack
X1 = hstack((categories_one_hot_train, sub_categories_one_hot_train, X_train_state_ohe, X_t
X2 = hstack((categories_one_hot_test, sub_categories_one_hot_test, X_test_state_ohe, X_test
X3 = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, X_cv_state_ohe, X_cv_teacher
print("Final Data matrix")
print(X1.shape, y_train.shape)
print(X3.shape, y_cv.shape)
print(X2.shape, y_test.shape)
```

```
Final Data matrix
(26275, 11227) (26275,)
(11261, 11227) (11261,)
(16088, 11227) (16088,)
```

In [153]:

```
X1 = X1.todense()
```

In [154]:

```
clf = DecisionTreeClassifier(random_state=0,class_weight='balanced')
clf = clf.fit(X1,y_train)
```

In [155]:

```
#https://stackoverflow.com/questions/49170296/scikit-learn-feature-importance-calculation-i
a = clf.tree_.compute_feature_importances(normalize=False)
```

In [156]:

```
prob_class = np.argsort((clf.tree_.compute_feature_importances(normalize=False)))
print(prob_class[:5000])
```

```
[   0 7185 7186 ..., 9325 9326 9327]
```

In [157]:

```
all_features = []
all_features.extend(vectorizer1.get_feature_names())
all_features.extend(vectorizer2.get_feature_names())
all_features.extend(vectorizer3.get_feature_names())
all_features.extend(vectorizer4.get_feature_names())
all_features.extend(vectorizer5)
all_features.extend(vectorizer6.get_feature_names())
all_features.extend(vectorizer7.get_feature_names())
all_features.append('price')
all_features.append('quantity')
all_features.append('teacher_number_of_previously_posted_projects')
```

In [158]:

```
np.take(all_features,prob_class[:5000])
```

Out[158]:

```
array(['Warmth', 'random', 'range', ..., 'unlikely', 'unlimited', 'unlock'],
      dtype='<U44')
```

In [159]:

```
df_train = pd.DataFrame(X1)
prob_class = prob_class[:5000]
df_train = df_train.iloc[:, prob_class]
df_train.shape
```

Out[159]:

```
(26275, 5000)
```

In [160]:

```
X2 = X2.todense()
df_test = pd.DataFrame(X2)
df_test = df_test.iloc[:, prob_class]
df_test.shape
```

Out[160]:

```
(16088, 5000)
```

In [161]:

```
X3 = X3.todense()
df_cv = pd.DataFrame(X3)
df_cv = df_cv.iloc[:, prob_class]
df_cv.shape
```

Out[161]:

```
(11261, 5000)
```

In [177]:

```
%%time
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.ht
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

d_tree = DecisionTreeClassifier(class_weight='balanced')
parameters = {'max_depth':[5, 10, 50, 100, 200, 500, 1000], 'min_samples_split': [ 5, 10,20
clf = GridSearchCV(d_tree, parameters, cv=3, scoring='roc_auc',return_train_score=True)
clf.fit(df_train, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
```

```
Wall time: 12h 59min 56s
```

In [178]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['train_auc'] = train_auc
df.head(5)
```

Out[178]:

	max_depth	min_split	train_auc
0	5	5	0.535220
1	10	10	0.535000
2	50	20	0.534817
3	100	30	0.534259
4	200	50	0.534183

In [179]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[179]:

		train_auc
max_depth	**min_split**	
5	**5**	0.535220
	10	0.576782
	20	0.759464
	30	0.841784
	50	0.892458
	80	0.922074
10	**5**	0.999064
	10	0.535000
	20	0.574754
	30	0.750641

In [180]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[180]:

	max_depth	min_split	train_auc
0	5	5	0.535220
1	5	10	0.576782
2	5	20	0.759464
3	5	30	0.841784
4	5	50	0.892458

In [181]:

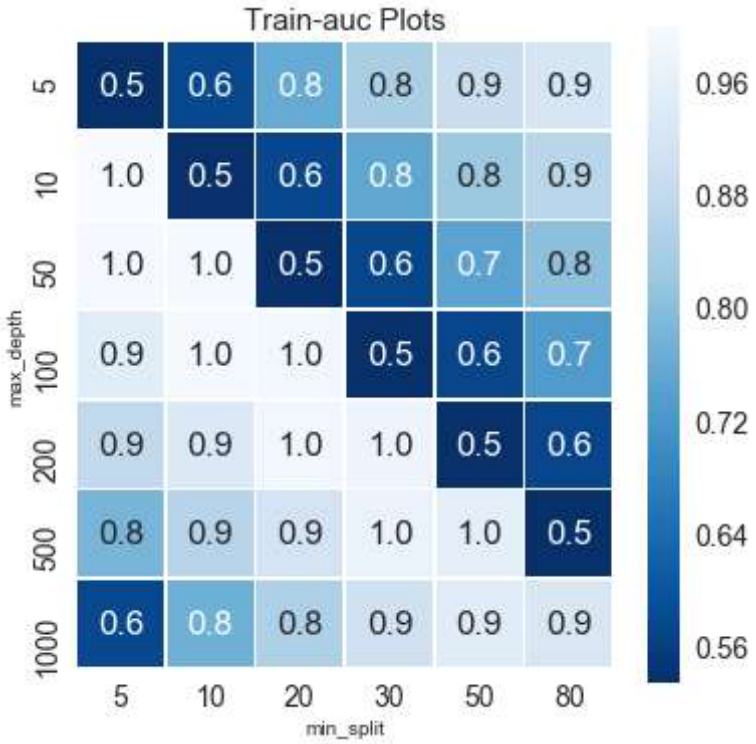
```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','train_auc').head()
```

Out[181]:

min_split	5	10	20	30	50	80
max_depth						
5	0.535220	0.576782	0.759464	0.841784	0.892458	0.922074
10	0.999064	0.535000	0.574754	0.750641	0.822149	0.864750
50	0.995889	0.995428	0.534817	0.572607	0.743135	0.804456
100	0.947168	0.992344	0.985669	0.534259	0.571720	0.728765
200	0.878109	0.934899	0.983506	0.975088	0.534183	0.570710

In [182]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','train_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Train-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```



In [183]:

```
max_depth = [5, 10, 50, 100, 200, 500, 1000, 5, 10, 50, 100, 200, 500, 1000 , 5, 10, 50, 10
min_sample_split = [ 5, 10,20,30,50,80, 5, 10,20,30,50,80,5, 10,20,30,50,80,5, 10,20,30,50,


df = pd.DataFrame()
df['max_depth'] = max_depth
df['min_split'] = min_sample_split
df['cv_auc'] = cv_auc
df.head(5)
```

Out[183]:

	max_depth	min_split	cv_auc
0	5	5	0.522637
1	10	10	0.522649
2	50	20	0.522594
3	100	30	0.522242
4	200	50	0.522177

In [184]:

```
phase_1_2 = df.groupby(['max_depth', 'min_split']).mean()
print (phase_1_2.shape)
phase_1_2.head(10)
```

(42, 1)

Out[184]:

		cv_auc
max_depth	**min_split**	
5	**5**	0.522637
	10	0.534338
	20	0.546729
	30	0.551006
	50	0.551685
	80	0.551925
10	**5**	0.540335
	10	0.522649
	20	0.534437
	30	0.549189

In [185]:

```
phase_1_2 = phase_1_2.reset_index()
phase_1_2.head()
```

Out[185]:

	max_depth	min_split	cv_auc
0	5	5	0.522637
1	5	10	0.534338
2	5	20	0.546729
3	5	30	0.551006
4	5	50	0.551685

In [186]:

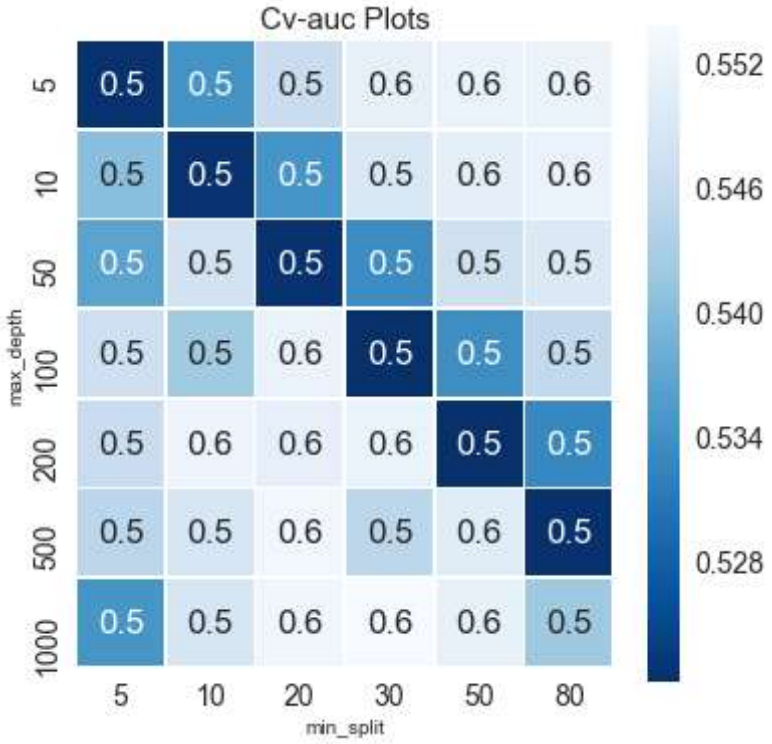
```
import numpy as np;
import seaborn as sns;

# To translate into Excel Terms for those familiar with Excel
# string 1 is row labels 'helix1 phase'
# string 2 is column labels 'helix 2 phase'
# string 3 is values 'Energy'
# Official pivot documentation
# http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.pivot.html

phase_1_2.pivot('max_depth', 'min_split','cv_auc').head()
```

Out[186]:

min_split	5	10	20	30	50	80
max_depth						
5	0.522637	0.534338	0.546729	0.551006	0.551685	0.551925
10	0.540335	0.522649	0.534437	0.549189	0.550820	0.551590
50	0.536523	0.547962	0.522594	0.533486	0.547429	0.549361
100	0.547134	0.542048	0.551602	0.522242	0.533546	0.545405
200	0.546413	0.552263	0.551125	0.551631	0.522177	0.532820

In [187]:

```
plt.figure(figsize=(6,6))
pivot_table = phase_1_2.pivot('max_depth', 'min_split','cv_auc')
plt.xlabel('max_depth', size = 10)
plt.ylabel('min_split', size = 10)
plt.title('Cv-auc Plots', size = 15)
sns.heatmap(pivot_table, annot=True, fmt=".1f", linewidths=.5, square = True, cmap = 'Blues
```

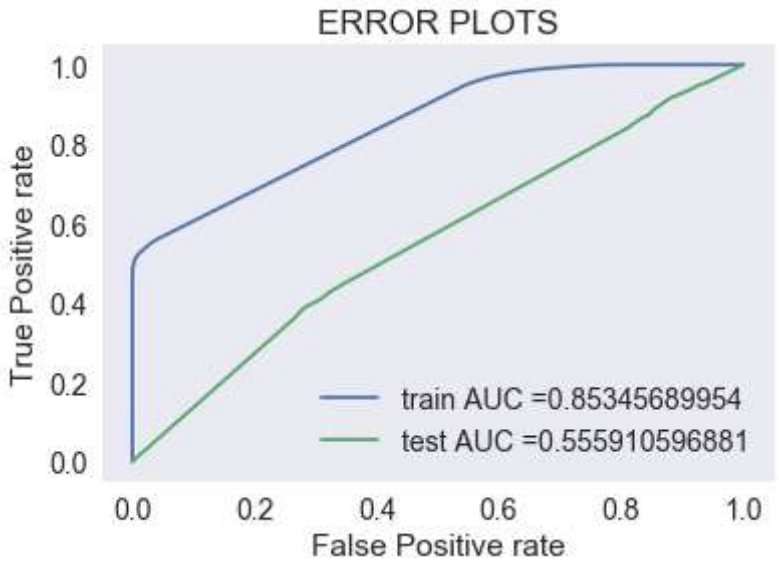


Optimal Parameters

min_sample_split = 20
max_depth = 100

In [193]:

```
from sklearn.metrics import roc_curve, auc


clf= DecisionTreeClassifier( min_samples_split = 20, max_depth = 100 , class_weight = 'bala
clf.fit(df_train, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs

y_train_pred = batch_predict(clf, df_train)
y_test_pred = batch_predict(clf, df_test)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

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("False Positive rate")
plt.ylabel("True Positive rate")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



In [194]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

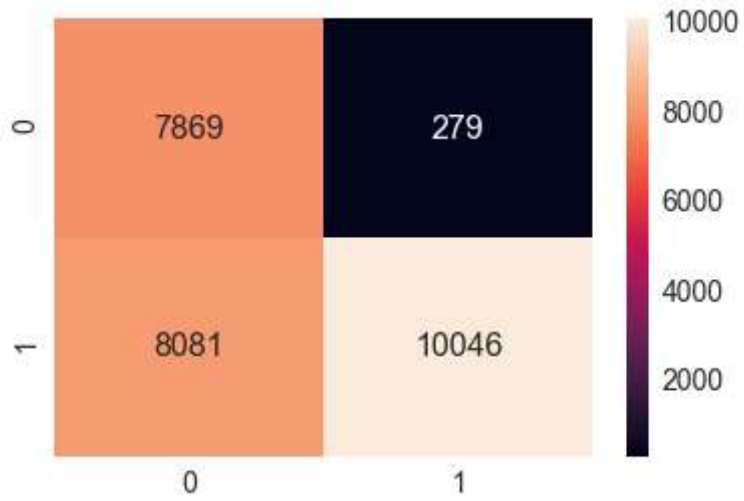
In [195]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
#https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

y_pred_new = clf.predict(df_train)

print("Confusion Matrix of train set:\n [ [TN  FP]\n [FN TP] ]\n")
df_cm = pd.DataFrame(confusion_matrix(y_train, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
====================================================================================
========================
Confusion Matrix of train set:
 [ [TN  FP]
 [FN TP] ]
```

Out[195]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x209a128df60>
```



In [196]:

```
y_pred_new = clf.predict(df_test)

df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred_new), range(2),range(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[196]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x209a02f6828>
```

3. Conclusion

In [197]:

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

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytab

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyperparameters(max depth,min samples split)", "Tr
x.add_row(["BOW", "Decision Trees","(10, 30)", 0.7244, 0.6197])
x.add_row(["TF-idf", "Decision Trees","(10, 30)", 0.7355, 0.6296])
x.add_row(["Avg-W2v", "Decision Trees","(10, 30)", 0.7355, 0.6299])
x.add_row(["Tfidf-w2v", "Decision Trees","(10, 30)", 0.7899, 0.5788])
x.add_row(["5K features", "Decision Trees","(1000, 20)", 0.8534, 0.5559])
print(x)
```

```
+-------------+----------------+-------------------------------------------
--+-----------+----------+
|  Vectorizer |     Model      | Hyperparameters(max depth,min samples spli
t) | Train AUC | Test AUC |
+-------------+----------------+-------------------------------------------
--+-----------+----------+
|     BOW     | Decision Trees |                  (10, 30)
|   0.7244  |  0.6197  |
|    TF-idf   | Decision Trees |                  (10, 30)
|   0.7355  |  0.6296  |
|   Avg-W2v   | Decision Trees |                  (10, 30)
|   0.7355  |  0.6299  |
|  Tfidf-w2v  | Decision Trees |                  (10, 30)
|   0.7899  |  0.5788  |
| 5K features | Decision Trees |                 (1000, 20)
|   0.8534  |  0.5559  |
+-------------+----------------+-------------------------------------------
--+-----------+----------+
```