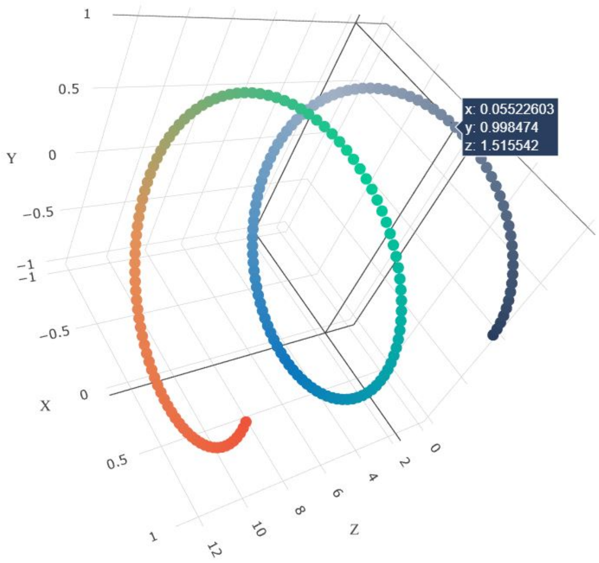
Assignment 8: DT

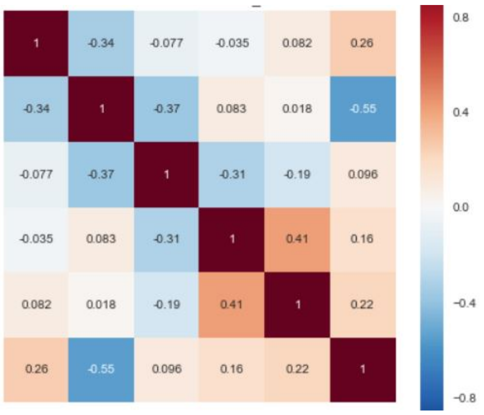
1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**
   - Set 1: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
   - Set 2: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)
2. **The hyper paramter tuning (best `depth` in range [1, 5, 10, 50], and the best `min_samples_split` in range [5, 10, 100, 500])**
   - Find the best hyper parameter which will give the maximum AUC value
   - find the best hyper paramter using k-fold cross validation(use gridsearch cv or randomsearch cv)/simple cross validation data(you can write your own for loops refer sample solution)
3. **Representation of results**
   - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



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

or

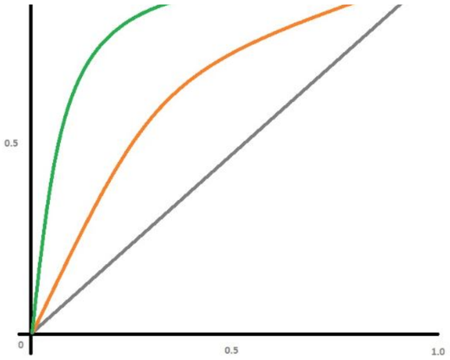
- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



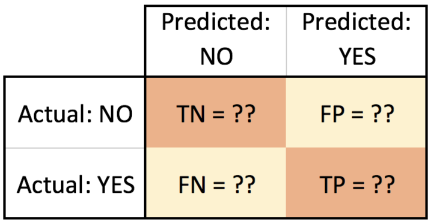
1	-0.34	-0.077	-0.035	0.082	0.26
-0.34	1	-0.37	0.083	0.018	-0.55
-0.077	-0.37	1	-0.31	-0.19	0.096
-0.035	0.083	-0.31	1	0.41	0.16
0.082	0.018	-0.19	0.41	1	0.22
0.26	-0.55	0.096	0.16	0.22	1

seaborn heat maps with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**

- You choose either of the plotting techniques out of 3d plot or heat map
- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.



- Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points

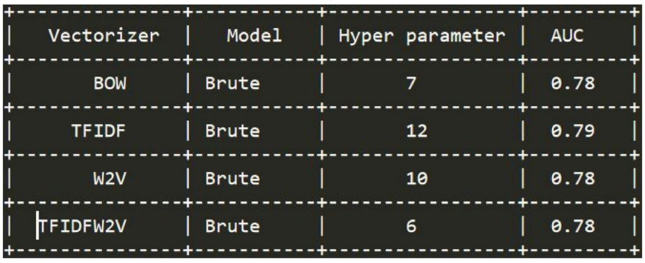


	Predicted: NO	Predicted: YES
Actual: NO	TN = ??	FP = ??
Actual: YES	FN = ??	TP = ??

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

4. **Task 2:** For this task consider set-1 features. Select all the features which are having non-zero feature importance.You can get the feature importance using 'feature_importances_` (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html), discard the all 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
Note: when you want to find the feature importance make sure you don't use max_depth parameter keep it None.

5. You need to summarize the results at the end of the notebook, summarize it in the table format



```
+------------+---------+-----------------+------+
| Vectorizer |  Model  | Hyper parameter | AUC  |
+------------+---------+-----------------+------+
|    BOW     | Brute   |        7        | 0.78 |
+------------+---------+-----------------+------+
|   TFIDF    | Brute   |        12       | 0.79 |
+------------+---------+-----------------+------+
|    W2V     | Brute   |        10       | 0.78 |
+------------+---------+-----------------+------+
| TFIDFW2V   | Brute   |        6        | 0.78 |
+------------+---------+-----------------+------+
```

1. Decision Tree

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

1.1 Loading Data

In [2]:

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

In [3]:

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

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

In [4]:

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

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

Out[4]:

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

2. Preprocessing

2.1 preprocessing of project_subject_categories

2.1 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/47301924/4084039

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

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

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

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

2.2 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/47301924/4084039

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

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

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

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

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

2.3 Text preprocessing of essay

In [7]:

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

In [8]:

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

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10

In [9]:

```
# 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 s
chool. \r\n\r\n We have over 24 languages represented in our English Learner program with students at e
very level of mastery.  We also have over 40 countries represented with the families within our school.
Each student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, bel
iefs, and respect.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein  Ou
r English learner's have a strong support system at home that begs for more resources.  Many times our
parents are learning to read and speak English along side of their children.  Sometimes this creates ba
rriers for parents to be able to help their child learn phonetics, letter recognition, and other readin
g skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of t
he English language even if no one at home is able to assist.  All families with students within the Le
vel 1 proficiency status, will be a offered to be a part of this program.  These educational videos wil
l be specially chosen by the English Learner Teacher and will be sent home regularly to watch.  The vid
eos are to help the child develop early reading skills.\r\n\r\nParents that do not have access to a dvd
player will have the opportunity to check out a dvd player to use for the year.  The plan is to use the
se videos and educational dvd's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, at least
The [illegible] fifth grade students that will cycle through my classroom this year all love learning, at least
most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560
students, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get togethe
r and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes tha
t students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, an
d games. At the end of the year the school hosts a carnival to celebrate the hard work put in during th
e school year, with a dunk tank being the most popular activity.My students will use these five brightl
y colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of
ten in the classroom and not enough for each student to have an individual one, they will be used in a
variety of ways. During independent reading time they will be used as special chairs students will each
use on occasion. I will utilize them in place of chairs at my small group tables during math and readin
g times. The rest of the day they will be used by the students who need the highest amount of movement
in their life in order to stay focused on school.\r\n\r\nWhenever asked what the classroom is missing,
my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. Wh
en the students are sitting in group with me on the Hokki Stools, they are always moving, but at the sa
me time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the
first to be taken. There are always students who head over to the kidney table to get one of the stools
who are disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hou
rs a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the
same time. These stools will help students to meet their 60 minutes a day of movement by allowing them
to activate their core muscles for balance while they sit. For many of my students, these chairs will t
ake away the barrier that exists in schools for a child who can't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desk
s, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to c
reate a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nMy class i
s made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I
school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our s
chool is an \"open classroom\" concept, which is very unique as there are no walls separating the class
rooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all t
he information and experiences and keep on wanting more.With these resources such as the comfy red thro
w pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help creat
e the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom en
vironment is very important in the success in each and every child's education. The nautical photo prop
s will be used with each child as they step foot into our classroom for the first time on Meet the Teac
her evening. I'll take pictures of each child with them, have them developed, and then hung in our clas
sroom ready for their first day of 4th grade.  This kind gesture will set the tone before even the firs
t day of school! The nautical thank you cards will be used throughout the year by the students as they
create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make o
ur classroom a fun, inviting, 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 helping with this project to make
our new school year a very successful one. Thank you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive de
lays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardes
t working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students
. I teach in a Title I school where most of the students receive free or reduced price lunch.  Despite
their disabilities and limitations, my students love coming to school and come eager to learn and explo
re.Have you ever felt like you had ants in your pants and you needed to groove and move as you were in
a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they
say.Wobble chairs are the answer and I love 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 don't want to s
it and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the ke
y to our success. The number toss and color and shape mats can make that happen. My students will forge
t they are doing work and just have the fun a 6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great tea
cher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-Amer
ican, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2%
African-American students. Most of the students are on free or reduced lunch. We aren't receiving docto
rs, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspirin
g minds of young children and we focus not only on academics but one smart, effective, efficient, and d
isciplined students with good character.In our classroom we can utilize the Bluetooth for swift transit
ions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due t
o the volume of my speaker my students can't hear videos or books clearly and it isn't making the lesso
ns as meaningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause
and replay it at any time.\r\nThe cart will allow me to have more room for storage of things that are n
eeded for the day and has an extra part to it I can use.  The table top chart has all of the letter, wo
rds and pictures for students to learn about different letters and it is more accessible.nannan
==================================================
```

In [10]:

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

def decontracted(phrase):
```

```
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 [11]:

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

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

In [12]:

```
# \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 language delays, cognitive de
lays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardes
t working past their limitations.     The materials we have are the ones I seek out for my students. I
teach in a Title I school where most of the students receive free or reduced price lunch.  Despite thei
r disabilities and limitations, my students love coming to school and come eager to learn and explore.H
ave you ever felt like you had ants in your pants and you needed to groove and move as you were in a me
eting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.
Wobble chairs are the answer and I love then because they develop their core, which enhances gross moto
r and in Turn fine motor skills.   They also want to learn through games, my kids do not want to sit an
d 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 the
y are doing work and just have the fun a 6 year old deserves.nannan
```

In [13]:

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

```
My kindergarten students have varied disabilities ranging from speech and language delays cognitive del
ays gross fine motor delays to autism They are eager beavers and always strive to work their hardest wo
rking past their limitations The materials we have are the ones I seek out for my students I teach in a
Title I school where most of the students receive free or reduced price lunch Despite their disabilitie
s and limitations my students love coming to school and come eager to learn and explore Have you ever f
elt like you had ants in your pants and you needed to groove and move as you were in a meeting This is
how my kids feel all the time The want to be able to move as they learn or so they say Wobble chairs ar
e the answer and I love then because they develop their core which enhances gross motor and in Turn fin
e motor skills They also want to learn through games my kids do not want to sit and do worksheets They
```

```
want to learn to count by jumping and playing Physical engagement is the key to our success The number
toss and color and shape mats can make that happen My students will forget they are doing work and just
have the fun a 6 year old deserves nannan
```

In [14]:

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

In [15]:

```
# 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%|██████████████████████████████████████████████████| 109248/109248 [01:36<00:
00, 1135.36it/s]
```

In [16]:

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

Out[16]:

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

In [17]:

In [17]:

```
project_data['clean_essay'] = preprocessed_essays
project_data.drop(['essay'], axis=1, inplace=True)
project_data.head(2)
```

Out[17]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10

2.4 Preprocessing of `project_title`

In [18]:

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

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
```

In [19]:

```
# 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 [20]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar

# https://gist.github.com/sebleier/554280

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)
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████| 109248/109248 [00:04<00:0
0, 24421.49it/s]
```

In [21]:

```
# after preprocesing
preprocessed_titles[20000]
```

Out[21]:

```
'need move input'
```

In [22]:

```
project_data['clean_project_title'] = preprocessed_titles
project_data.drop(['project_title'], axis=1, inplace=True)
project_data.head(2)
```

Out[22]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10

2.5 Cleaning data of project_grade_category

In [23]:

```
#cleaning project_grade_category

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
grade_list = []
for i in grades:
    i = i.replace('-','_')
    i = i.replace(' ','')
    grade_list.append(i)
```

In [24]:

```
project_data['clean_grade_category'] = grade_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data.head(2)
```

Out[24]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10

2.6 Droping unnecessary columns

In [25]:

```
#project_data.drop(['id'], axis=1, inplace=True)
project_data.drop(['teacher_id'], axis=1, inplace=True)
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
project_data.drop(['project_resource_summary'], axis=1, inplace=True)
project_data.drop(['Unnamed: 0'], axis=1, inplace=True)
project_data.head(2)
```

Out[25]:

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec
0	p253737	Mrs.	IN	2016-12-05 13:43:57	0
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

2.7 Adding price column in our dataframe

In [26]:

```
resource_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1541272 entries, 0 to 1541271
Data columns (total 4 columns):
```

```
Data columns (total 4 columns):
id             1541272 non-null object
description    1540980 non-null object
quantity       1541272 non-null int64
price          1541272 non-null float64
dtypes: float64(1), int64(1), object(2)
memory usage: 47.0+ MB
```

In [27]:

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

Out[27]:

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec
0	p253737	Mrs.	IN	2016-12-05 13:43:57	0
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

In [28]:

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

In [29]:

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

Out[29]:

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec
0	p253737	Mrs.	IN	2016-12-05 13:43:57	0
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

2.8 Adding quantity column in our dataframe

In [30]:

```
resource_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1541272 entries, 0 to 1541271
Data columns (total 4 columns):
id             1541272 non-null object
description    1540980 non-null object
quantity       1541272 non-null int64
price          1541272 non-null float64
dtypes: float64(1), int64(1), object(2)
memory usage: 47.0+ MB
```

In [31]:

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

Out[31]:

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec
0	p253737	Mrs.	IN	2016-12-05 13:43:57	0
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

In [32]:

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

In [33]:

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

Out[33]:

	id	teacher_prefix	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projec
0	p253737	Mrs.	IN	2016-12-05 13:43:57	0
1	p258326	Mr.	FL	2016-10-25 09:22:10	7

2.9 Preprocessing of teacher_prefix

In [34]:

```
import re
prefix = list(project_data['teacher_prefix'].values)

prefix_list = []

for i in prefix:

    j=str(i)
    j=j.lower()
    j = re.sub(r"\.", "",j)

    prefix_list.append(j)


#print(prefix_list)
```

In [35]:

```
project_data['clean_teacher_prefix'] = prefix_list
project_data.drop(['teacher_prefix'], axis=1, inplace=True)
project_data.head(2)
```

Out[35]:

	id	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projects	project_is_a
0	p253737	IN	2016-12-05 13:43:57	0	0
1	p258326	FL	2016-10-25 09:22:10	7	1

2.10 Preprocessing of school_state

In [36]:

```
state = list(project_data['school_state'].values)

state_list = []

for i in state:

    j=str(i)
    j=j.lower()


    state_list.append(j)


#print(state_list)
```

In [37]:

```
project_data['clean_school_state'] = state_list
#project_data.drop(['school_state'], axis=1, inplace=True)
project_data.head(2)
```

Out[37]:

	id	school_state	project_submitted_datetime	teacher_number_of_previously_posted_projects	project_is_a
0	p253737	IN	2016-12-05 13:43:57	0	0
1	p258326	FL	2016-10-25 09:22:10	7	1

3. Decision Tree

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

In [38]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use


from sklearn.model_selection import train_test_split

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn import model_selection


X = project_data.drop(['project_is_approved','id'], axis=1)
X.head(2)

y = project_data['project_is_approved'].values


# split the data set into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,shuffle=False)

# split the train data set into cross validation train and cross validation test
#X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.2,shuffle=False)

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

```
(87398, 12) (87398,)
(21850, 12) (21850,)
```

3.2 Make Data Model Ready: encoding numerical, categorical features

3.2.1 encoding categorical features: School State

In [39]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_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 = vectorizer.transform(X_train['clean_school_state'].values)
#X_cv_state_ohe = vectorizer.transform(X_cv['clean_school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['clean_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(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(87398, 51) (87398,)
(21850, 51) (21850,)
['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']
====================================================================================================
```

3.2.2 encoding categorical features: teacher prefix

In [40]:

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer.transform(X_train['clean_teacher_prefix'].values)
#X_cv_teacher_ohe = vectorizer.transform(X_cv['clean_teacher_prefix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['clean_teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
#print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(87398, 6) (87398,)
(21850, 6) (21850,)
['dr', 'mr', 'mrs', 'ms', 'nan', 'teacher']
====================================================================================================
```

3.2.3 encoding categorical features: project_grade_category

In [41]:

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer.transform(X_train['clean_grade_category'].values)
#X_cv_grade_ohe = vectorizer.transform(X_cv['clean_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['clean_grade_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
#print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(87398, 4) (87398,)
(21850, 4) (21850,)
['grades3_5', 'grades6_8', 'grades9_12', 'gradesprek_2']
====================================================================================================
```

3.2.4 encoding categorical features: project_subject_categories

In [42]:

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_categories_ohe = vectorizer.transform(X_train['clean_categories'].values)
#X_cv_categories_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_categories_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_categories_ohe.shape, y_train.shape)
#print(X_cv_categories_ohe.shape, y_cv.shape)
```

```
print(X_test_categories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(87398, 9) (87398,)
(21850, 9) (21850,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language', 'math_scienc
e', 'music_arts', 'specialneeds', 'warmth']
====================================================================================================
```

3.2.5 encoding categorical features: project_subject_subcategories

In [43]:

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_subcategories_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
#X_cv_subcategories_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_subcategories_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_subcategories_ohe.shape, y_train.shape)
#print(X_cv_subcategories_ohe.shape, y_cv.shape)
print(X_test_subcategories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(87398, 30) (87398,)
(21850, 30) (21850,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government', 'college_careerprep', 'co
mmunityservice', 'earlydevelopment', 'economics', 'environmentalscience', 'esl', 'extracurricular', 'fi
nancialliteracy', 'foreignlanguages', 'gym_fitness', 'health_lifescience', 'health_wellness', 'history_
geography', 'literacy', 'literature_writing', 'mathematics', 'music', 'nutritioneducation', 'other', 'p
arentinvolvement', 'performingarts', 'socialsciences', 'specialneeds', 'teamsports', 'visualarts', 'war
mth']
====================================================================================================
```

3.2.6 encoding numerical feature: price

In [44]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
# 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.
scaler.fit(X_train['price'].values.reshape(-1,1))

X_train_price_scaler = scaler.transform(X_train['price'].values.reshape(-1,1))
X_test_price_scaler = scaler.transform(X_test['price'].values.reshape(-1,1))




print("After vectorizations")
print(X_train_price_scaler.shape, y_train.shape)
print(X_test_price_scaler.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(87398, 1) (87398,)
(21850, 1) (21850,)
====================================================================================================
```

In [45]:

```
print(X_train_price_scaler)
```

```
[[-3.87742480e-01]
 [ 5.98715095e-04]
 [ 5.86472187e-01]
 ...
 [-4.08584892e-01]
 [-3.19460050e-01]
 [-7.65891064e-01]]
```

3.2.7 encoding numerical feature: quantity

In [46]:

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

X_train_quantity_scaler = scaler.transform(X_train['quantity'].values.reshape(-1,1))
X_test_quantity_scaler = scaler.transform(X_test['quantity'].values.reshape(-1,1))




print("After vectorizations")
print(X_train_quantity_scaler.shape, y_train.shape)
print(X_test_quantity_scaler.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(87398, 1) (87398,)
(21850, 1) (21850,)
====================================================================================================
```

3.2.8 encoding numerical feature: teacher_number_of_previously_posted_projects

In [47]:

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

X_train_posted_project_scaler = scaler.transform(X_train['teacher_number_of_previously_posted_projects'
].values.reshape(-1,1))
#X_cv_posted_project_norm = normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].v
alues.reshape(1,-1))
X_test_posted_project_scaler = scaler.transform(X_test['teacher_number_of_previously_posted_projects'].
values.reshape(-1,1))

# X_train_posted_project_scaler = X_train_posted_project_scaler.reshape(-1,1)
# #X_cv_posted_project_norm = X_cv_posted_project_norm.reshape(-1,1)
# X_test_posted_project_scaler = X_test_posted_project_scaler.reshape(-1,1)


print("After vectorizations")
print(X_train_posted_project_scaler.shape, y_train.shape)
#print(X_cv_posted_project_norm.shape, y_cv.shape)
print(X_test_posted_project_scaler.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(87398, 1) (87398,)
(21850, 1) (21850,)
====================================================================================================
```

3.3 Make Data Model Ready: encoding eassay, and project_title

3.3.1 encoding essay

3.3.1.1 encoding essay : TFIDF

In [48]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(project_data['clean_essay'].values)

X_train_essay_tfidf = vectorizer.transform(X_train['clean_essay'].values)
#X_cv_essay_tfidf= vectorizer.transform(X_cv['clean_essay'].values)
X_test_essay_tfidf = vectorizer.transform(X_test['clean_essay'].values)

print("After vectorizations")
print(X_train_essay_tfidf.shape, y_train.shape)
#print(X_cv_essay_tfidf.shape, y_cv.shape)
print(X_test_essay_tfidf.shape, y_test.shape)
#print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(87398, 16623) (87398,)
(21850, 16623) (21850,)
====================================================================================================
```

3.3.1.2 encoding essay : TFIDF W2V

In [49]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-an
d-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [50]:

```
# Similarly you can vectorize for essay

tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['clean_essay'].values)
# 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 [51]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
essay_tfidf_w2v_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_essay'].values): # 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
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_train.append(vector)

print(len(essay_tfidf_w2v_train))
print(len(essay_tfidf_w2v_train[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 87398/87398 [04:29<00
:00, 324.30it/s]
```

```
87398
300
```

In [52]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
essay_tfidf_w2v_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    essay_tfidf_w2v_test.append(vector)

print(len(essay_tfidf_w2v_test))
print(len(essay_tfidf_w2v_test[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 21850/21850 [01:11<00
:00, 303.51it/s]
```

```
21850
300
```

3.3.2 encoding titles

3.3.2.1 encoding titles : TFIDF

In [53]:

```
vectorizer = TfidfVectorizer(min_df=10)
vectorizer.fit(project_data['clean_project_title'].values)

X_train_title_tfidf = vectorizer.transform(X_train['clean_project_title'].values)
#X_cv_title_tfidf= vectorizer.transform(X_cv['clean_project_title'].values)
X_test_title_tfidf = vectorizer.transform(X_test['clean_project_title'].values)

print("After vectorizations")
print(X_train_title_tfidf.shape, y_train.shape)
#print(X_cv_title_tfidf.shape, y_cv.shape)
print(X_test_title_tfidf.shape, y_test.shape)
print(X_test_title_tfidf.shape, y_test.shape)
#print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(87398, 3222) (87398,)
(21850, 3222) (21850,)
====================================================================================================
```

3.3.2.2 encoding titles : TFIDF W2V

In [54]:

```
# Similarly you can vectorize for title also

tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['clean_project_title'].values)
# 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 [55]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
title_tfidf_w2v_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['clean_project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    title_tfidf_w2v_train.append(vector)

print(len(title_tfidf_w2v_train))
print(len(title_tfidf_w2v_train[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 87398/87398 [00:04<00:0
0, 21715.43it/s]
```

```
87398
300
```

In [56]:

```
# tfidf Word2Vec
# compute tfidf word2vec for each review.
title_tfidf_w2v_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['clean_project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word
)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf
value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    title_tfidf_w2v_test.append(vector)
```

```
print(len(title_tfidf_w2v_test))
print(len(title_tfidf_w2v_test[0]))
```

```
100%|████████████████████████████████████████████████████████████| 21850/21850 [00:01<00:0
0, 20683.55it/s]
```

```
21850
300
```

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

3.4.1 Applying Decision Tree on TFIDF, SET 1

In [110]:

```
# Please write all the code with proper documentation

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr_tfidf = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_categories_ohe,
X_train_subcategories_ohe, X_train_price_scaler, X_train_quantity_scaler, X_train_posted_project_scaler
, X_train_essay_tfidf, X_train_title_tfidf)).tocsr()
X_te_tfidf = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categories_ohe, X_t
est_subcategories_ohe, X_test_price_scaler, X_test_quantity_scaler, X_test_posted_project_scaler, X_tes
t_essay_tfidf, X_test_title_tfidf)).tocsr()

y_train_tfidf = y_train
y_test_tfidf = y_test

print("Final Data matrix")
print(X_tr_tfidf.shape, y_train_tfidf.shape)
print(X_te_tfidf.shape, y_test_tfidf.shape)
print("="*100)
```

```
Final Data matrix
(87398, 19948) (87398,)
(21850, 19948) (21850,)
====================================================================================================
```

3.4.1.1 Hyperparameter Tuning

In [111]:

```
from sklearn import tree
from sklearn.model_selection import GridSearchCV

clf_tfidf = tree.DecisionTreeClassifier(criterion='gini', class_weight='balanced' )
tuned_parameters = {'max_depth':[1,5,10,50],'min_samples_split':[5,10,100,500]}

model_tfidf = GridSearchCV(clf_tfidf, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,return_
train_score=True)
model_tfidf.fit(X_tr_tfidf, y_train_tfidf)

print(model_tfidf.best_estimator_)
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   13.9s
[Parallel(n_jobs=-1)]: Done  48 out of  48 | elapsed: 10.2min finished
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=10,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [112]:

```
import matplotlib.pyplot as plt


train_auc= model_tfidf.cv_results_['mean_train_score']
cv_auc = model_tfidf.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
min_samples_split = tuned_parameters['min_samples_split']
print(min_samples_split)
```

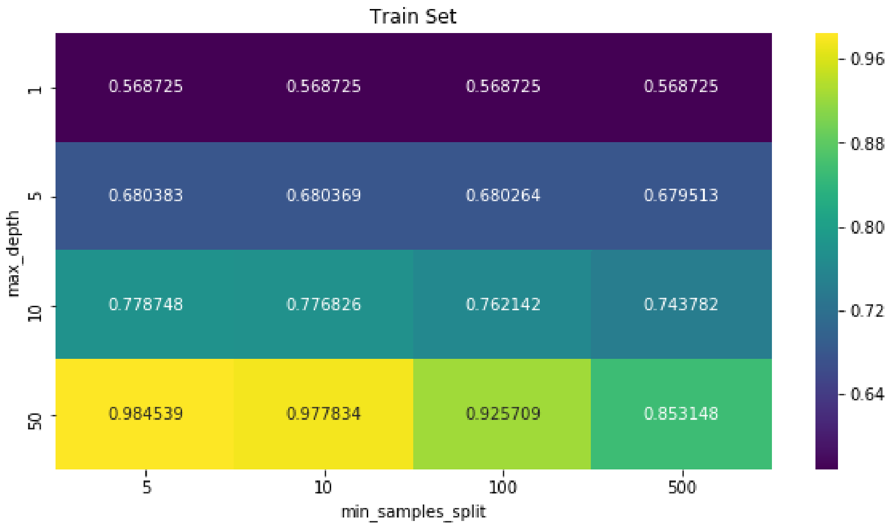
```
[1, 5, 10, 50]
[5, 10, 100, 500]
```

In [113]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=min_samples_split,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=min_samples_split,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

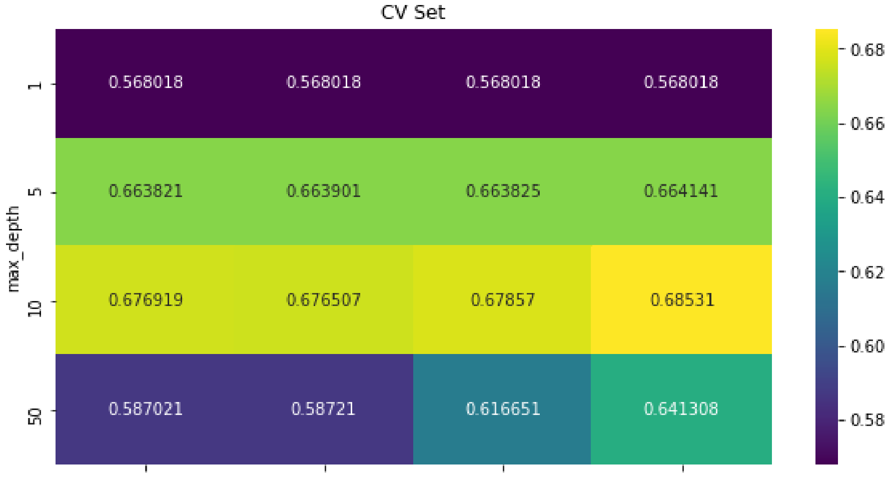
In [114]:

```
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

param_max_depth = model_tfidf.cv_results_['param_max_depth']
param_min_samples_split = model_tfidf.cv_results_['param_min_samples_split']

plt.figure(figsize = (10,5))
df = pd.DataFrame({'max_depth': param_max_depth, 'min_samples_split': param_min_samples_split, 'train_a
uc':train_auc})
result = df.pivot(index='max_depth', columns='min_samples_split', values='train_auc')
sns.heatmap(result, annot=True, fmt="g", cmap='viridis')
plt.title("Train Set")


plt.show()
```



In [115]:

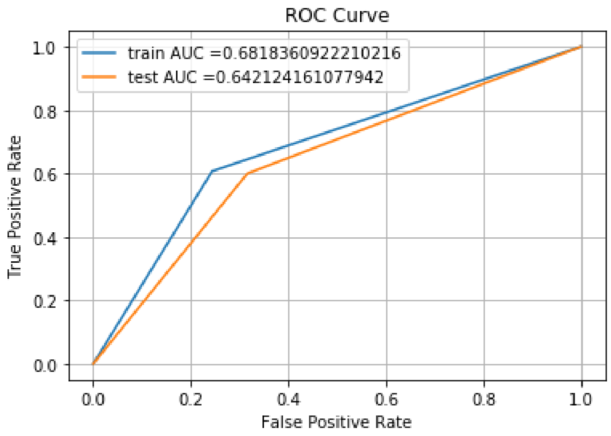
```
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

param_max_depth = model_tfidf.cv_results_['param_max_depth']
param_min_samples_split = model_tfidf.cv_results_['param_min_samples_split']

plt.figure(figsize = (10,5))
df = pd.DataFrame({'max_depth': param_max_depth, 'min_samples_split': param_min_samples_split, 'cv_auc'
:cv_auc})
result = df.pivot(index='max_depth', columns='min_samples_split', values='cv_auc')
sns.heatmap(result, annot=True, fmt="g", cmap='viridis')
plt.title("CV Set")


plt.show()
```



3.4.1.2 Testing the performance of the model on test data, plotting ROC Curves

In [116]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_
curve
from sklearn.metrics import roc_curve, auc




clf_tfidf = tree.DecisionTreeClassifier(max_depth=10,min_samples_split=500,criterion='gini',class_weigh
t='balanced')
clf_tfidf.fit(X_tr_tfidf, y_train_tfidf)






y_train_pred = clf_tfidf.predict(X_tr_tfidf)
y_test_pred = clf_tfidf.predict(X_te_tfidf)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_tfidf, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_tfidf, 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("ROC Curve")
plt.grid()
plt.show()
```



In [117]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [118]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.45939717020949306 for threshold 1
```

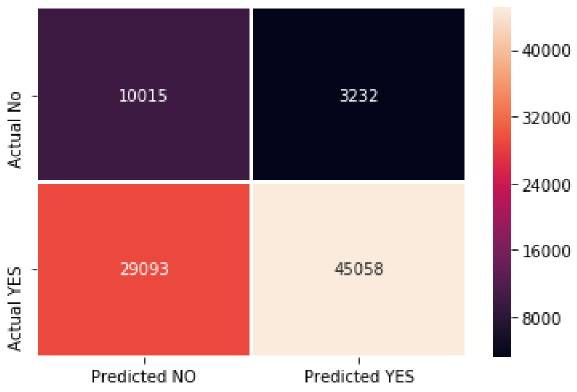
In [119]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [120]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t))
```

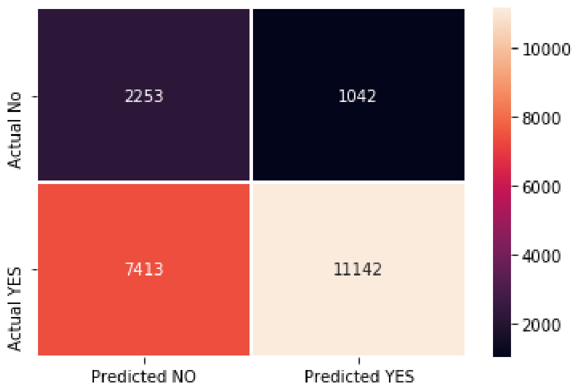
```
Train confusion matrix
```



In [121]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```



3.4.1.3 Visualizing Decision Tree

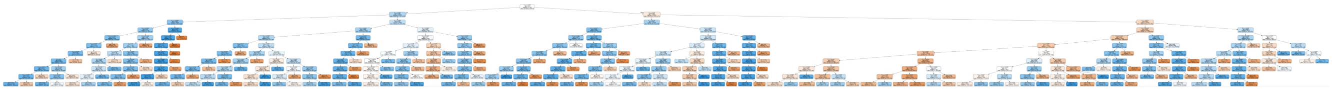
In [122]:

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

# import graphviz
# from sklearn import tree


# tfidf_dt = tree.export_graphviz(clf_tfidf,filled=True, rounded=True, special_characters=True,rotate=T
rue)
# graph = graphviz.Source(tfidf_dt)
# graph.render("tfidf_dt")   #saving in pdf file
# graph
```

In [123]:

```
#https://github.com/bhattbhavesh91/visualize-decision-tree/blob/master/visualize-dt-notebook.ipynb
import numpy as np
import pandas as pd
from pandas import DataFrame, Series
from IPython.display import Image
import io
import pydotplus
from sklearn import preprocessing
from sklearn import tree
%matplotlib inline

def plot_decision_tree(clf):
    dot_data = io.StringIO()
    tree.export_graphviz(clf, out_file=dot_data,

                         filled=True, rounded=True,
                         special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return Image(graph.create_png())


plot_decision_tree(clf_tfidf)
```

Out[123]:



3.4.1.4 False Point Datapoints Analysis

In [125]:

```
y_test_pred.shape
```

Out[125]:

```
(21850,)
```

In [126]:

```
type(y_test.shape[0])
```

Out[126]:

```
int
```

In [127]:

```
false_positive_index = []

false_positive_datapoints = []
for i in range(y_test.shape[0]):
```

```
for i in range(y_test.shape[0]):
    if((y_test[i]==0 ) and (y_test_pred[i]==1)):
        false_positive_index.append(i)
```

3.4.1.4.1 Word Cloud on False Point Datapoints of clean_essay

In [128]:

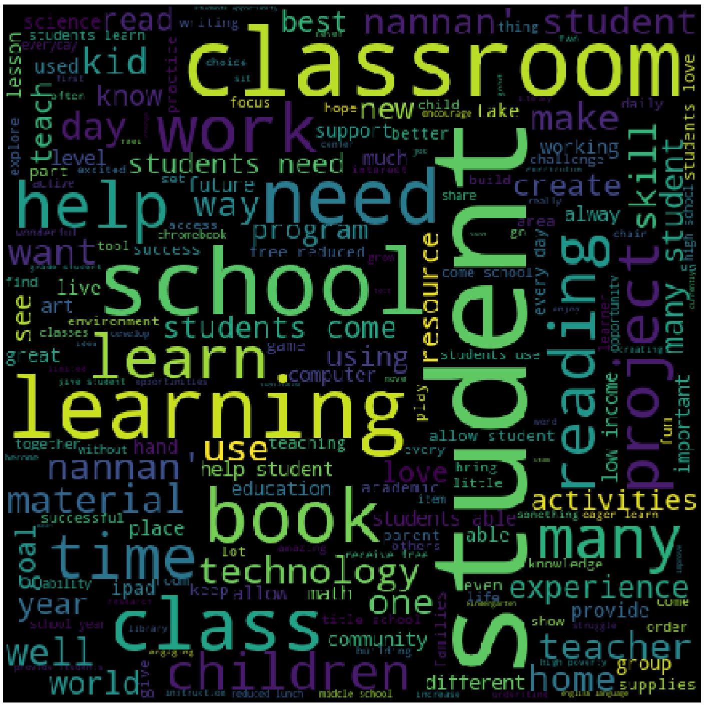
```
for i in false_positive_index:
    false_positive_datapoints.append(X_test['clean_essay'].values[i])

len(false_positive_datapoints)
```

Out[128]:

```
1042
```

In [129]:

```
from wordcloud import WordCloud
from PIL import Image

#wine_mask = np.array(Image.open("wine.jpg"))

wordcloud = WordCloud(background_color="black",width=500,height=500).generate(str(false_positive_datapo
ints))

plt.figure(figsize=(8,8))
plt.imshow(wordcloud)
#plt.imshow(wine_mask)
plt.axis('off')
plt.show()
```



3.4.1.4.2 Boxplot on False Point Datapoints of price

In [130]:

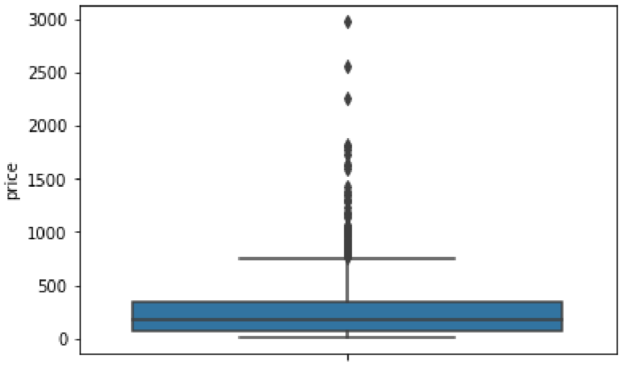
```
price_fp = []
for i in false_positive_index:
    price_fp.append(X_test['price'].values[i])

price_df = pd.DataFrame({'price':price_fp})
```

In [131]:

```
sns.boxplot(y='price',data=price_df)
```

Out[131]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x256e05ac7b8>
```



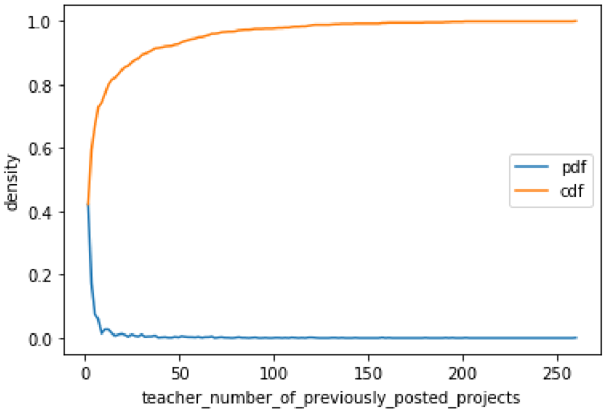
3.4.1.4.3 PDF & CDF on False Point Datapoints of teacher_number_of_previously_posted_projects

In [132]:

```
teacher_prev_posted = []
for i in false_positive_index:
    teacher_prev_posted.append(X_test['teacher_number_of_previously_posted_projects'].values[i])
```

In [133]:

```
import matplotlib.pyplot as plt

counts, bin_edges = np.histogram(teacher_prev_posted,bins='auto',density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)


plt.xlabel("teacher_number_of_previously_posted_projects")
plt.ylabel("density")


plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)

label = ["pdf","cdf"]

plt.legend(label)

plt.show()
```



3.4.2 Applying Decision Tree on TFIDF W2V, SET 2

In [134]:

```
# Please write all the code with proper documentation

# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr_tfidfw2v = hstack((X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_categories_o
he, X_train_subcategories_ohe, X_train_price_scaler, X_train_posted_project_scaler, X_train_quantity_sc
aler, essay_tfidf_w2v_train, title_tfidf_w2v_train)).tocsr()
X_te_tfidfw2v = hstack((X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe, X_test_categories_ohe,
X_test_subcategories_ohe, X_test_price_scaler, X_test_quantity_scaler, X_test_posted_project_scaler,ess
ay_tfidf_w2v_test, title_tfidf_w2v_test)).tocsr()

y_train_tfidfw2v = y_train
y_test_tfidfw2v = y_test

print("Final Data matrix")
print(X_tr_tfidfw2v.shape, y_train_tfidfw2v.shape)
print(X_te_tfidfw2v.shape, y_test_tfidfw2v.shape)
print("="*100)
```

```
Final Data matrix
(87398, 703) (87398,)
(21850, 703) (21850,)
====================================================================================================
```

3.4.2.1 Hyperparameter Tuning

In [135]:

```
from sklearn import tree
from sklearn.model_selection import GridSearchCV

clf_tfidfw2v = tree.DecisionTreeClassifier(criterion='gini', class_weight='balanced' )
tuned_parameters = {'max_depth':[1,5,10,50],'min_samples_split':[5,10,100,500]}

model_tfidfw2v = GridSearchCV(clf_tfidfw2v, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,r
eturn_train_score=True)
model_tfidfw2v.fit(X_tr_tfidfw2v, y_train_tfidfw2v)

print(model_tfidfw2v.best_estimator_)
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:   49.9s
[Parallel(n_jobs=-1)]: Done  48 out of  48 | elapsed: 31.2min finished
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=5,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [136]:

```
import matplotlib.pyplot as plt


train_auc= model_tfidfw2v.cv_results_['mean_train_score']
cv_auc = model_tfidfw2v.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
min_samples_split = tuned_parameters['min_samples_split']
```

```
print(min_samples_split)
```

```
[1, 5, 10, 50]
[5, 10, 100, 500]
```

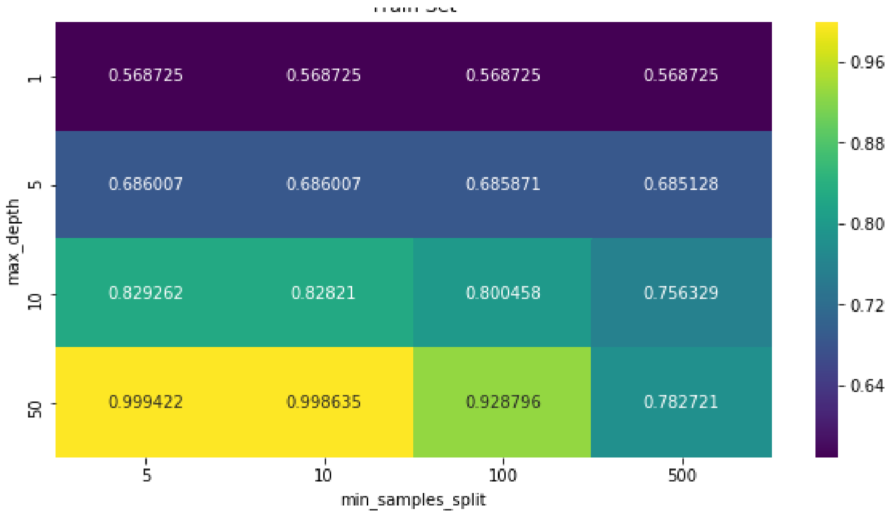
In [137]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=min_samples_split,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=min_samples_split,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

In [138]:

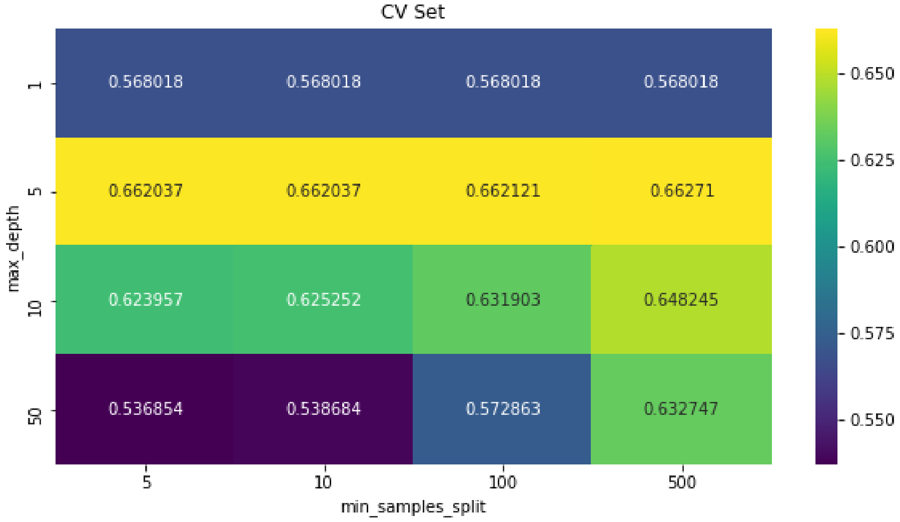
```
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

param_max_depth = model_tfidfw2v.cv_results_['param_max_depth']
param_min_samples_split = model_tfidfw2v.cv_results_['param_min_samples_split']

plt.figure(figsize = (10,5))
df = pd.DataFrame({'max_depth': param_max_depth, 'min_samples_split': param_min_samples_split, 'train_a
uc':train_auc})
result = df.pivot(index='max_depth', columns='min_samples_split', values='train_auc')
sns.heatmap(result, annot=True, fmt="g", cmap='viridis')
plt.title("Train Set")


plt.show()
```

Train Set



In [139]:

```
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

param_max_depth = model_tfidfw2v.cv_results_['param_max_depth']
param_min_samples_split = model_tfidfw2v.cv_results_['param_min_samples_split']

plt.figure(figsize = (10,5))
df = pd.DataFrame({'max_depth': param_max_depth, 'min_samples_split': param_min_samples_split, 'cv_auc'
:cv_auc})
result = df.pivot(index='max_depth', columns='min_samples_split', values='cv_auc')
sns.heatmap(result, annot=True, fmt="g", cmap='viridis')
plt.title("CV Set")


plt.show()
```

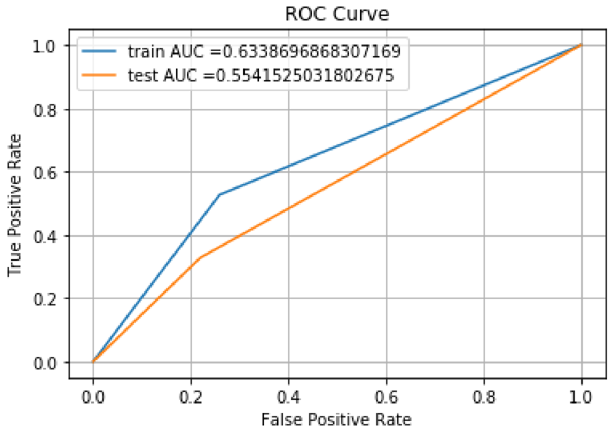


3.4.2.2 Testing the performance of the model on test data, plotting ROC Curves

In [140]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_
curve
from sklearn.metrics import roc_curve, auc
```

```
clf_tfidfw2v = tree.DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=5,
                      min_samples_split=500)
clf_tfidfw2v.fit(X_tr_tfidfw2v, y_train_tfidfw2v)







y_train_pred = clf_tfidfw2v.predict(X_tr_tfidfw2v)
y_test_pred = clf_tfidfw2v.predict(X_te_tfidfw2v)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_tfidfw2v, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_tfidfw2v, 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("ROC Curve")
plt.grid()
plt.show()
```



In [141]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [142]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.3903304808535335 for threshold 1
```

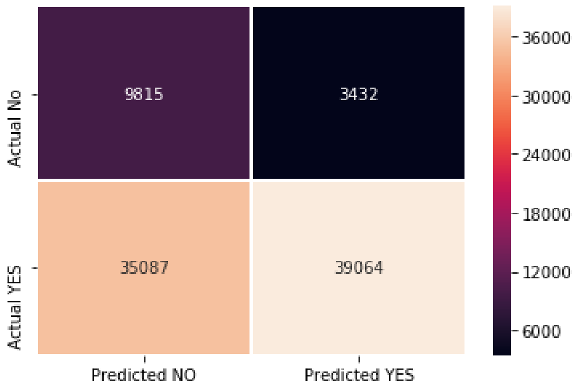
In [143]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [144]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t))
```

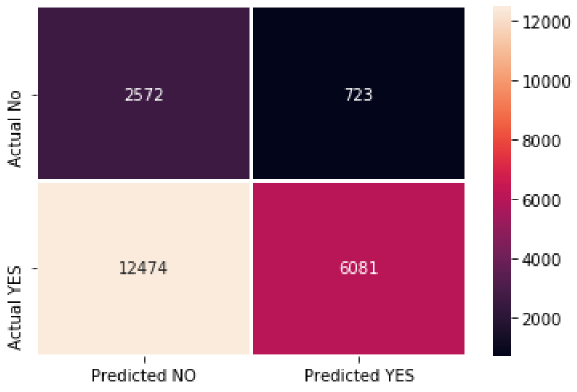
```
Train confusion matrix
```



In [145]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```

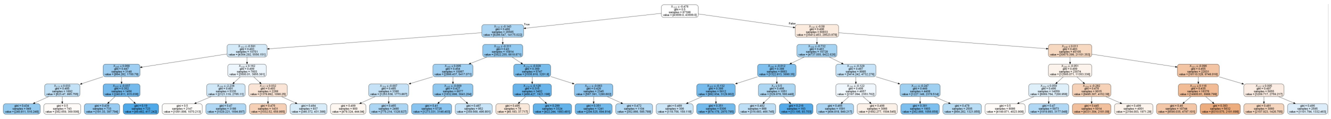


3.4.2.3 Visualizing Decision Tree

In [146]:

```
#https://github.com/bhattbhavesh91/visualize-decision-tree/blob/master/visualize-dt-notebook.ipynb
import numpy as np
import pandas as pd
from pandas import DataFrame, Series
from IPython.display import Image
import io
import pydotplus
from sklearn import preprocessing
from sklearn import tree
%matplotlib inline

def plot_decision_tree(clf):
    dot_data = io.StringIO()
    tree.export_graphviz(clf, out_file=dot_data,
    tree.export_graphviz(clf, out_file=dot_data,
                         
                         filled=True, rounded=True,
                         special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return Image(graph.create_png())


plot_decision_tree(clf_tfidfw2v)
```

Out[146]:



3.4.1.4 False Point Datapoints Analysis

In [147]:

```
y_test_pred.shape
```

Out[147]:

```
(21850,)
```

In [148]:

```
type(y_test.shape[0])
```

Out[148]:

```
int
```

In [149]:

```
false_positive_index = []

false_positive_datapoints = []
for i in range(y_test.shape[0]):
    if((y_test[i]==0 ) and (y_test_pred[i]==1)):
          false_positive_index.append(i)
```

3.4.2.4.1 Word Cloud on False Point Datapoints of clean_essay

In [150]:

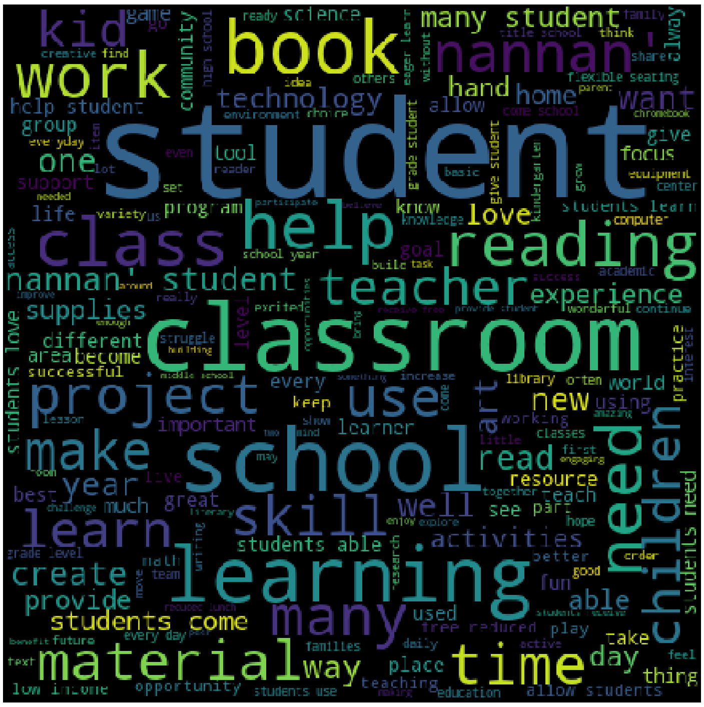
```
for i in false_positive_index:
    false_positive_datapoints.append(X_test['clean_essay'].values[i])

len(false_positive_datapoints)
```

Out[150]:

```
723
```

In [151]:

```
from wordcloud import WordCloud
from PIL import Image

#wine_mask = np.array(Image.open("wine.jpg"))

wordcloud = WordCloud(background_color="black",width=500,height=500).generate(str(false_positive_datapo
ints))

plt.figure(figsize=(8,8))
plt.figure(figsize=(8,8))
plt.imshow(wordcloud)
#plt.imshow(wine_mask)
plt.axis('off')
plt.show()
```



3.4.2.4.2 Boxplot on False Point Datapoints of price

In [152]:

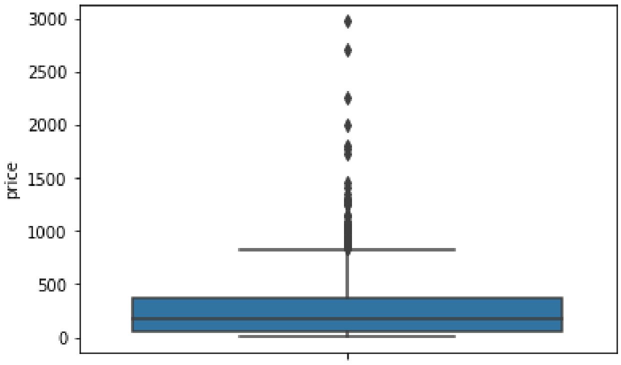
```
price_fp = []
for i in false_positive_index:
    price_fp.append(X_test['price'].values[i])

price_df = pd.DataFrame({'price':price_fp})
```

In [153]:

```
sns.boxplot(y='price',data=price_df)
```

Out[153]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x256858e1cf8>
```



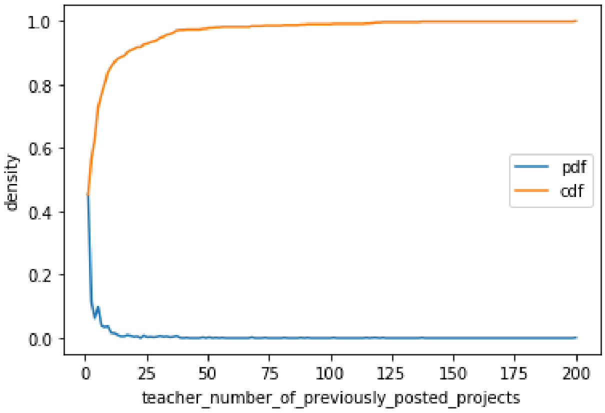
3.4.2.4.3 PDF & CDF on False Point Datapoints of teacher_number_of_previously_posted_projects

In [154]:

```
teacher_prev_posted = []
for i in false_positive_index:
    teacher_prev_posted.append(X_test['teacher_number_of_previously_posted_projects'].values[i])
```

In [155]:

```
import matplotlib.pyplot as plt

counts, bin_edges = np.histogram(teacher_prev_posted,bins='auto',density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)


plt.xlabel("teacher_number_of_previously_posted_projects")
plt.ylabel("density")


plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)

label = ["pdf","cdf"]

plt.legend(label)

plt.show()
```



3.4.3 Applying Decision Tree on TFIDF (having non zero feature importances), SET 3

In [156]:

```
index=[]
for i in range(len(clf_tfidf.feature_importances_)):
    if clf_tfidf.feature_importances_[i]>0:
        index.append(i)
```

In [157]:

```
len(index)
```

Out[157]:

```
136
```

In [158]:

```
print(index)
```

```
[59, 65, 100, 101, 102, 147, 248, 354, 577, 773, 894, 1054, 1359, 1383, 1464, 1491, 1566, 1614, 1685, 1
715, 1993, 2495, 2639, 2644, 2768, 2813, 2886, 2900, 3564, 3642, 3841, 3870, 3940, 3976, 4369, 4702, 48
23, 4999, 5013, 5261, 5327, 5382, 5889, 5993, 6145, 6255, 6276, 6390, 6602, 6809, 6862, 6864, 7056, 714
8, 7233, 7256, 7258, 7271, 7629, 7669, 7829, 7982, 7992, 8073, 8132, 8133, 8414, 8457, 8664, 8665, 8667
, 8668, 8725, 8733, 8738, 8742, 8930, 8962, 9193, 9245, 9422, 9491, 9596, 9766, 9843, 9850, 9879, 9951
, 8668, 8725, 8733, 8738, 8742, 8930, 8962, 9193, 9245, 9422, 9491, 9596, 9766, 9843, 9850, 9879, 9931,
9958, 10076, 10109, 10652, 10696, 10783, 10842, 11012, 11285, 11346, 11540, 11580, 11650, 11838, 11854,
11876, 11907, 12106, 12110, 12489, 12781, 12887, 12909, 13214, 13270, 13289, 13620, 13931, 13936, 14280
, 14388, 14407, 14776, 14967, 15043, 15172, 15522, 15700, 15898, 15901, 15903, 16144, 16219, 16221, 163
45, 16514, 16676, 18379]
```

In [159]:

```
clf_tfidf.feature_importances_[3147]
```

Out[159]:

```
0.0
```

In [160]:

```
# top_tfidf_fi_index = clf_tfidf.feature_importances_.argsort()[::-1][:5000]
```

In [161]:

```
# len(top_tfidf_fi_index)
```

In [162]:

```
# X_train_set3 = X_tr_tfidf[:,top_tfidf_fi_index]
# y_train_set3 = y_train_tfidf

# X_test_set3 = X_te_tfidf[:,top_tfidf_fi_index]
# y_test_set3 = y_test_tfidf
```

In [163]:

```
X_train_set3 = X_tr_tfidf[:,index]
y_train_set3 = y_train_tfidf

X_test_set3 = X_te_tfidf[:,index]
y_test_set3 = y_test_tfidf
```

In [164]:

```
X_train_set3.shape

print("Final Data matrix")
print(X_train_set3.shape, y_train_set3.shape)
print(X_test_set3.shape, y_test_set3.shape)
print("="*100)
```

```
Final Data matrix
(87398, 136) (87398,)
(21850, 136) (21850,)
====================================================================================================
```

3.4.3.1 Hyperparameter Tuning

In [165]:

```
from sklearn import tree
from sklearn.model_selection import GridSearchCV

clf_tfidfset3 = tree.DecisionTreeClassifier(criterion='gini', class_weight='balanced' )
tuned_parameters = {'max_depth':[1,5,10,50],'min_samples_split':[5,10,100,500]}

model_tfidfset3 = GridSearchCV(clf_tfidf, tuned_parameters, scoring = 'roc_auc',verbose=5,n_jobs=-1,ret
urn_train_score=True)
model_tfidfset3.fit(X_train_set3, y_train_set3)

print(model_tfidfset3.best_estimator_)
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=-1)]: Done  10 tasks      | elapsed:    6.0s
[Parallel(n_jobs=-1)]: Done  48 out of  48 | elapsed:  1.6min finished
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=10,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

In [166]:

```
import matplotlib.pyplot as plt


train_auc= model_tfidfset3.cv_results_['mean_train_score']
cv_auc = model_tfidfset3.cv_results_['mean_test_score']

max_depth = tuned_parameters['max_depth']
print(max_depth)
min_samples_split = tuned_parameters['min_samples_split']
print(min_samples_split)
```

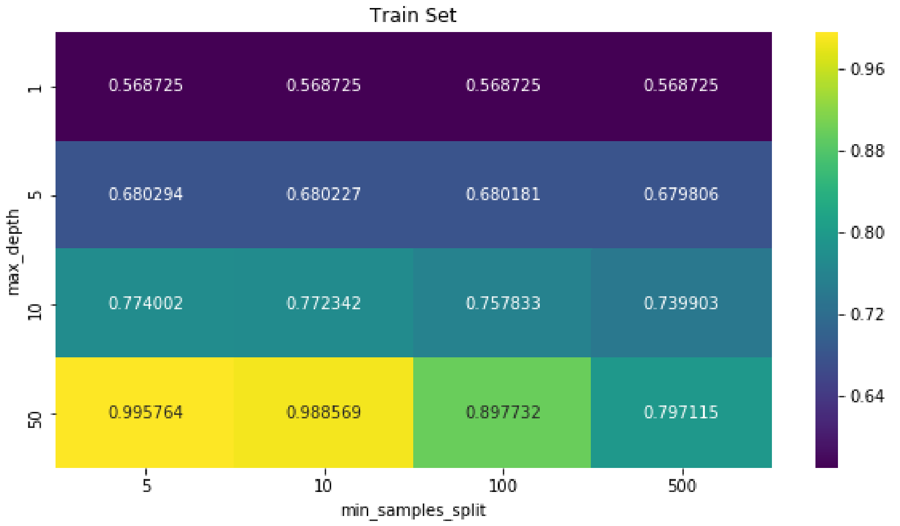
```
[1, 5, 10, 50]
[5, 10, 100, 500]
```

In [167]:

```
# https://plot.ly/python/3d-axes/
trace1 = go.Scatter3d(x=min_samples_split,y=max_depth,z=train_auc, name = 'train')
trace2 = go.Scatter3d(x=min_samples_split,y=max_depth,z=cv_auc, name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))

fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

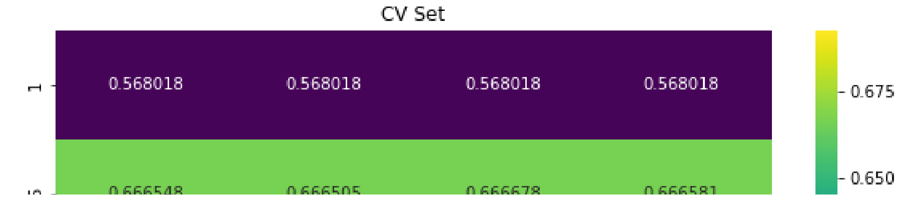
In [168]:

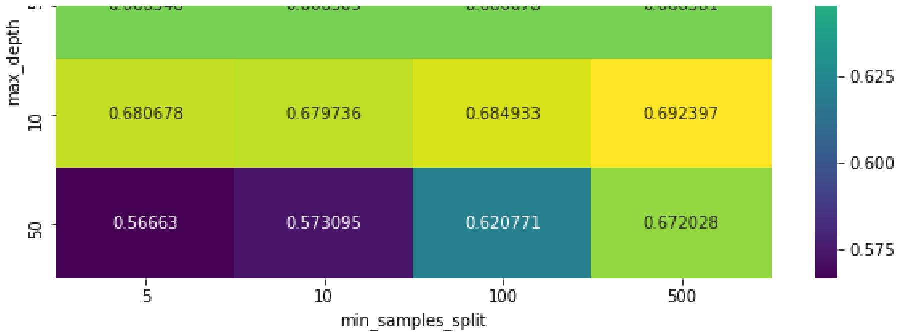
```
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

param_max_depth = model_tfidfset3.cv_results_['param_max_depth']
param_min_samples_split = model_tfidfset3.cv_results_['param_min_samples_split']

plt.figure(figsize = (10,5))
df = pd.DataFrame({'max_depth': param_max_depth, 'min_samples_split': param_min_samples_split, 'train_a
uc':train_auc})
result = df.pivot(index='max_depth', columns='min_samples_split', values='train_auc')
sns.heatmap(result, annot=True, fmt="g", cmap='viridis')
plt.title("Train Set")


plt.show()
```



In [169]:

```
#https://stackoverflow.com/questions/37790429/seaborn-heatmap-using-pandas-dataframe

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

param_max_depth = model_tfidfset3.cv_results_['param_max_depth']
param_min_samples_split = model_tfidfset3.cv_results_['param_min_samples_split']

plt.figure(figsize = (10,5))
df = pd.DataFrame({'max_depth': param_max_depth, 'min_samples_split': param_min_samples_split, 'cv_auc'
:cv_auc})
result = df.pivot(index='max_depth', columns='min_samples_split', values='cv_auc')
sns.heatmap(result, annot=True, fmt="g", cmap='viridis')
plt.title("CV Set")


plt.show()
```

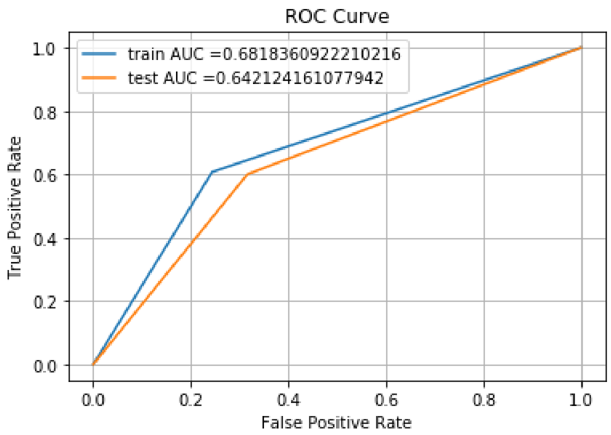




3.4.3.2 Testing the performance of the model on test data, plotting ROC Curves

In [170]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_
curve
from sklearn.metrics import roc_curve, auc




clf_tfidfset3 = tree.DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=10,
                       min_samples_split=500)
clf_tfidfset3.fit(X_train_set3, y_train_set3)






y_train_pred = clf_tfidfset3.predict(X_train_set3)
y_test_pred = clf_tfidfset3.predict(X_test_set3)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train_set3, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test_set3, 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("ROC Curve")
plt.grid()
plt.show()
```



In [171]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(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,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [172]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
```

```
====================================================================================================
the maximum value of tpr*(1-fpr) 0.45939717020949306 for threshold 1
```

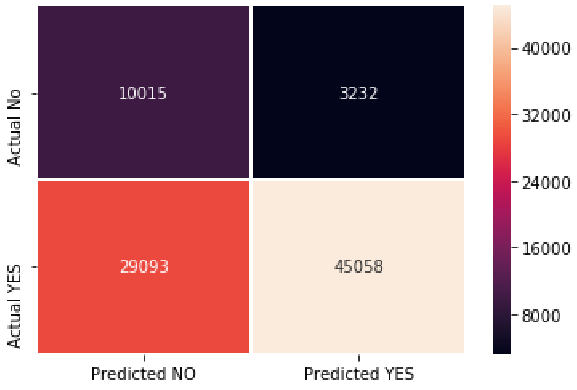
In [173]:

```
def get_confusion_matrix(y,y_pred):

    df = pd.DataFrame(confusion_matrix(y,y_pred),range(2),range(2))
    df.columns = ['Predicted NO','Predicted YES']
    df = df.rename({0:'  Actual No',1:'  Actual YES'})
    sns.heatmap(df,annot=True,fmt='g',linewidth=0.5)
```

In [174]:

```
print("Train confusion matrix")
get_confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t))
```

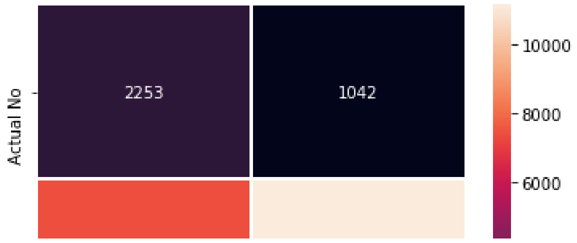
```
Train confusion matrix
```

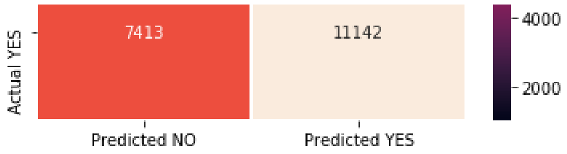


In [175]:

```
print("Test confusion matrix")
get_confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t))
```

```
Test confusion matrix
```



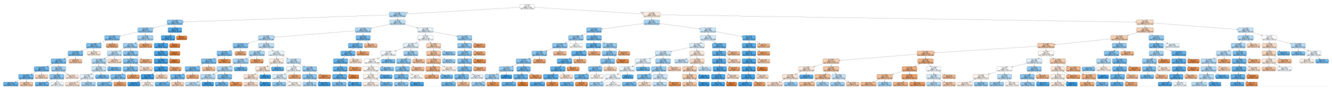


3.4.3.3 Visualizing Decision Tree

In [176]:

```
#https://github.com/bhattbhavesh91/visualize-decision-tree/blob/master/visualize-dt-notebook.ipynb
import numpy as np
import pandas as pd
from pandas import DataFrame, Series
from IPython.display import Image
import io
import pydotplus
from sklearn import preprocessing
from sklearn import tree
%matplotlib inline

def plot_decision_tree(clf):
    dot_data = io.StringIO()
    tree.export_graphviz(clf, out_file=dot_data,

                         filled=True, rounded=True,
                         special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
    return Image(graph.create_png())


plot_decision_tree(clf_tfidfset3)
```

Out[176]:



In [177]:

```
y_test_pred.shape
```

Out[177]:

```
(21850,)
```

In [178]:

```
type(y_test.shape[0])
```

Out[178]:

```
int
```

In [179]:

```
false_positive_index = []

false_positive_datapoints = []
for i in range(y_test.shape[0]):
    if((y_test[i]==0 ) and (y_test_pred[i]==1)):
          false_positive_index.append(i)
```

3.4.1.4.1 Word Cloud on False Point Datapoints of clean_essay

In [180]:

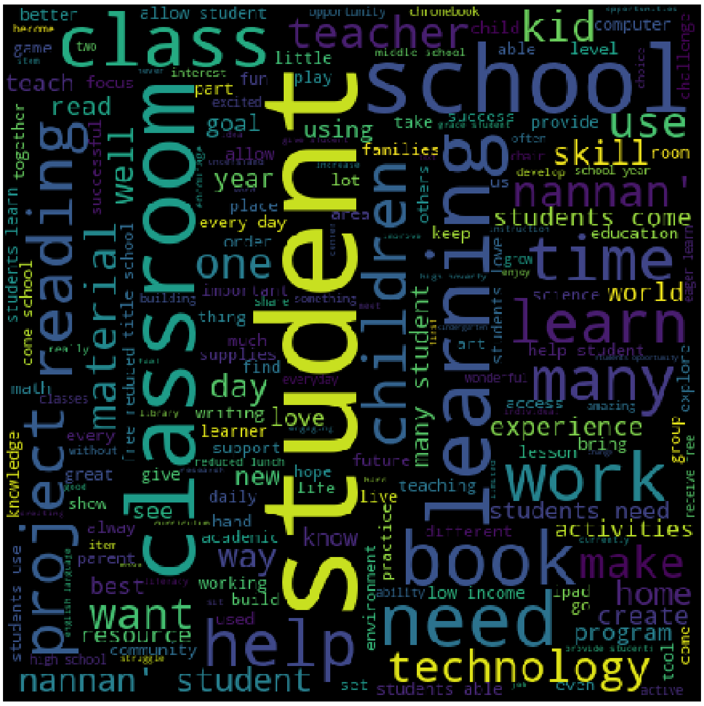
```
for i in false_positive_index:
    false_positive_datapoints.append(X_test['clean_essay'].values[i])

len(false_positive_datapoints)
```

Out[180]:

```
1042
```

In [181]:

```
from wordcloud import WordCloud
from PIL import Image

#wine_mask = np.array(Image.open("wine.jpg"))

wordcloud = WordCloud(background_color="black",width=500,height=500).generate(str(false_positive_datapo
ints))

plt.figure(figsize=(8,8))
plt.imshow(wordcloud)
#plt.imshow(wine_mask)
plt.axis('off')
plt.show()
```



3.4.3.4.2 Boxplot on False Point Datapoints of price

In [182]:

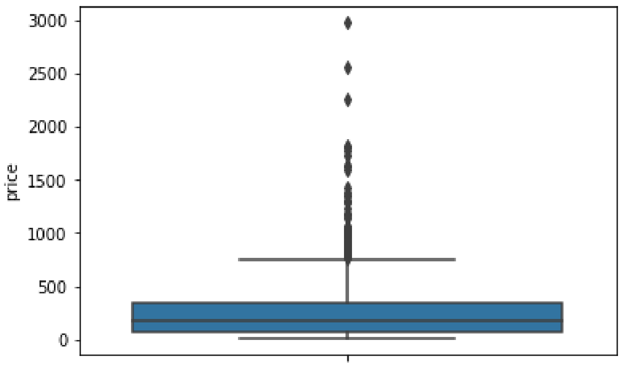
```
price_fp = []
for i in false_positive_index:
    price_fp.append(X_test['price'].values[i])

price_df = pd.DataFrame({'price':price_fp})
```

In [183]:

```
sns.boxplot(y='price',data=price_df)
```

Out[183]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x256dc34cb38>
```



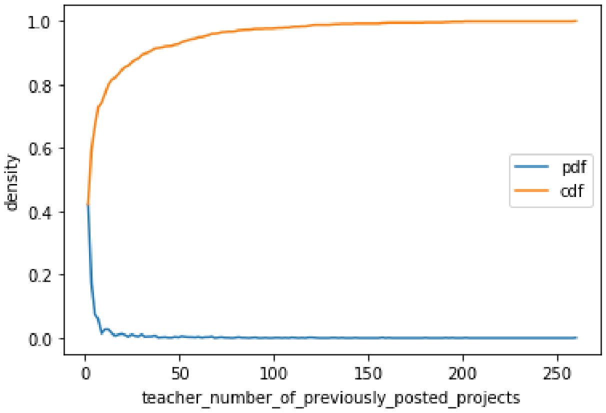
3.4.1.4.3 PDF & CDF on False Point Datapoints of teacher_number_of_previously_posted_projects

In [184]:

```
teacher_prev_posted = []
for i in false_positive_index:
    teacher_prev_posted.append(X_test['teacher_number_of_previously_posted_projects'].values[i])
```

In [185]:

```
import matplotlib.pyplot as plt

counts, bin_edges = np.histogram(teacher_prev_posted,bins='auto',density = True)
pdf = counts/(sum(counts))
cdf = np.cumsum(pdf)


plt.xlabel("teacher_number_of_previously_posted_projects")
plt.ylabel("density")


plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:],cdf)

label = ["pdf","cdf"]

plt.legend(label)

plt.show()
```



4. Conclusion

In [187]:

```
# Please compare all your models using Prettytable library
# http://zetcode.com/python/prettytable/
```

```python
from prettytable import PrettyTable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyperparameter(max_depth)", "Hyperparameter(min_sample_split)"
, "AUC"]


x.add_row(["TFIDF", "Decision Tree", 10, 500, 0.64212])
x.add_row(["TFIDF W2V", "Decision Tree", 5, 500, 0.55415])
x.add_row(["Set 3", "Decision Tree", 10, 500, 0.64212])


print(x)
```

```
+------------+---------------+---------------------------+----------------------------------+---------+
| Vectorizer |     Model     | Hyperparameter(max_depth) | Hyperparameter(min_sample_split) |   AUC   |
+------------+---------------+---------------------------+----------------------------------+---------+
|   TFIDF    | Decision Tree |             10            |               500                | 0.64212 |
| TFIDF W2V  | Decision Tree |             5             |               500                | 0.55415 |
|   Set 3    | Decision Tree |             10            |               500                | 0.64212 |
+------------+---------------+---------------------------+----------------------------------+---------+
```