DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Desc
`project_id`	A unique identifier for the proposed project. **Example:** `p0`
`project_title`	Title of the project. **Exa** • `Art Will Make You H` • `First Grad`
`project_grade_category`	Grade level of students for which the project is targeted. One of the fo enumerated v • `Grades P` • `Grade` • `Grade` • `Grades`
`project_subject_categories`	One or more (comma-separated) subject categories for the project fr following enumerated list of v • `Applied Lea` • `Care & H` • `Health & S` • `History & C` • `Literacy & Lan` • `Math & Sc` • `Music & The` • `Special` • `W` **Exar** • `Music & The` • `Literacy & Language, Math & Sc`

Feature	Desc
school_state	State where school is located (Two-letter U.S. post (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_c **Exampl**
project_subject_subcategories	One or more (comma-separated) subject subcategories for the **Exar** • `Lit` • `Literature & Writing, Social Sci`
project_resource_summary	An explanation of the resources needed for the project. **Exa** • `My students need hands on literacy materials to ma sensory needs!<`
project_essay_1	First application
project_essay_2	Second application
project_essay_3	Third application
project_essay_4	Fourth application
project_submitted_datetime	Datetime when project application was submitted. **Example:** `2016-( 12:43:5`
teacher_id	A unique identifier for the teacher of the proposed project. **Exa** `bdf8baa8fedef6bfeec7ae4ff1c`
teacher_prefix	Teacher's title. One of the following enumerated v • • • • • • `Tea`
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same te **Examp**

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

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

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

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

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

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

Notes on the Essay Data

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

- __project_essay_1:__ "Introduce us to your classroom"
- __project_essay_2:__ "Tell us more about your students"
- __project_essay_3:__ "Describe how your students will use the materials you're requesting"
- __project_essay_3:__ "Close by sharing why your project will make a difference"

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [1]:

```
import sys
!{sys.executable} -m pip install gensim
```

```
Collecting gensim
  Using cached https://files.pythonhosted.org/packages/ef/65/c90886ac34d4b12
d3ae0bcc7aece1af57e1f30e7138aabbb3e3c027e705a/gensim-3.8.0-cp35-cp35m-manyli
nux1_x86_64.whl (https://files.pythonhosted.org/packages/ef/65/c90886ac34d4b
12d3ae0bcc7aece1af57e1f30e7138aabbb3e3c027e705a/gensim-3.8.0-cp35-cp35m-many
linux1_x86_64.whl)
Collecting scipy>=0.18.1 (from gensim)
  Using cached https://files.pythonhosted.org/packages/7a/0e/3781e028d62a842
2244582abd8f084e6314297026760587c85607f687bf3/scipy-1.3.1-cp35-cp35m-manylin
ux1_x86_64.whl (https://files.pythonhosted.org/packages/7a/0e/3781e028d62a84
22244582abd8f084e6314297026760587c85607f687bf3/scipy-1.3.1-cp35-cp35m-manyli
nux1_x86_64.whl)
Collecting six>=1.5.0 (from gensim)
  Using cached https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1f
ecfe898238ce23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.wh
l (https://files.pythonhosted.org/packages/73/fb/00a976f728d0d1fecfe898238ce
23f502a721c0ac0ecfedb80e0d88c64e9/six-1.12.0-py2.py3-none-any.whl)
Collecting smart-open>=1.7.0 (from gensim)
Collecting numpy>=1.11.3 (from gensim)
  Using cached https://files.pythonhosted.org/packages/69/25/eef8d362bd216b1
1e7d005331a3cca3d19b0aa57569bde680070109b745c/numpy-1.17.0-cp35-cp35m-manyli
nux1_x86_64.whl (https://files.pythonhosted.org/packages/69/25/eef8d362bd216
b11e7d005331a3cca3d19b0aa57569bde680070109b745c/numpy-1.17.0-cp35-cp35m-many
linux1_x86_64.whl)
Collecting boto>=2.32 (from smart-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/23/10/c0b78c27298029e
4454a472a1919bde20cb182dab1662cec7f2ca1dcc523/boto-2.49.0-py2.py3-none-any.w
hl (https://files.pythonhosted.org/packages/23/10/c0b78c27298029e4454a472a19
19bde20cb182dab1662cec7f2ca1dcc523/boto-2.49.0-py2.py3-none-any.whl)
Collecting boto3 (from smart-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/ff/3e/2262936ad70fd6e
7b8827d79d6508ce33e2ffb49bfca6fedc5fe4abd6f1c/boto3-1.9.215-py2.py3-none-an
y.whl (https://files.pythonhosted.org/packages/ff/3e/2262936ad70fd6e7b8827d7
9d6508ce33e2ffb49bfca6fedc5fe4abd6f1c/boto3-1.9.215-py2.py3-none-any.whl)
Collecting requests (from smart-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/51/bd/23c926cd341ea6b
7dd0b2a00aba99ae0f828be89d72b2190f27c11d4b7fb/requests-2.22.0-py2.py3-none-a
ny.whl (https://files.pythonhosted.org/packages/51/bd/23c926cd341ea6b7dd0b2a
00aba99ae0f828be89d72b2190f27c11d4b7fb/requests-2.22.0-py2.py3-none-any.whl)
Collecting botocore<1.13.0,>=1.12.215 (from boto3->smart-open>=1.7.0->gensi
m)
  Using cached https://files.pythonhosted.org/packages/a1/b0/7a8794d914b95ef
3335a5a4ba20595b46081dbd1e29f13812eceacf091ca/botocore-1.12.215-py2.py3-none
-any.whl (https://files.pythonhosted.org/packages/a1/b0/7a8794d914b95ef3335a
5a4ba20595b46081dbd1e29f13812eceacf091ca/botocore-1.12.215-py2.py3-none-any.
whl)
Collecting s3transfer<0.3.0,>=0.2.0 (from boto3->smart-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/16/8a/1fc3dba0c4923c2
a76e1ff0d52b305c44606da63f718d14d3231e21c51b0/s3transfer-0.2.1-py2.py3-none-
any.whl (https://files.pythonhosted.org/packages/16/8a/1fc3dba0c4923c2a76e1f
f0d52b305c44606da63f718d14d3231e21c51b0/s3transfer-0.2.1-py2.py3-none-any.wh
l)
Collecting jmespath<1.0.0,>=0.7.1 (from boto3->smart-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/83/94/7179c3832a6d45b
266ddb2aac329e101367fbdb11f425f13771d27f225bb/jmespath-0.9.4-py2.py3-none-an
y.whl (https://files.pythonhosted.org/packages/83/94/7179c3832a6d45b266ddb2a
ac329e101367fbdb11f425f13771d27f225bb/jmespath-0.9.4-py2.py3-none-any.whl)
Collecting chardet<3.1.0,>=3.0.2 (from requests->smart-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e427
4b6487b4bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-an
y.whl (https://files.pythonhosted.org/packages/bc/a9/01ffebfb562e4274b6487b4
bb1ddec7ca55ec7510b22e4c51f14098443b8/chardet-3.0.4-py2.py3-none-any.whl)
Collecting urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 (from requests->smart-ope
n>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/e6/60/247f23a7121ae63
2d62811ba7f273d0e58972d75e58a94d329d51550a47d/urllib3-1.25.3-py2.py3-none-an
y.whl (https://files.pythonhosted.org/packages/e6/60/247f23a7121ae632d62811b
a7f273d0e58972d75e58a94d329d51550a47d/urllib3-1.25.3-py2.py3-none-any.whl)
Collecting idna<2.9,>=2.5 (from requests->smart-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/14/2c/cd551d81dbe1520
0be1cf41cd03869a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl
 (https://files.pythonhosted.org/packages/14/2c/cd551d81dbe15200be1cf41cd038
69a46fe7226e7450af7a6545bfc474c9/idna-2.8-py2.py3-none-any.whl)
Collecting certifi>=2017.4.17 (from requests->smart-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/69/1b/b853c7a9d4f6a6d
00749e94eb6f3a041e342a885b87340b79c1ef73e3a78/certifi-2019.6.16-py2.py3-none
-any.whl (https://files.pythonhosted.org/packages/69/1b/b853c7a9d4f6a6d00749
e94eb6f3a041e342a885b87340b79c1ef73e3a78/certifi-2019.6.16-py2.py3-none-any.
whl)
Collecting python-dateutil<3.0.0,>=2.1; python_version >= "2.7" (from botoco
re<1.13.0,>=1.12.215->boto3->smart-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/41/17/c62faccbfbd163c
7f57f3844689e3a78bae1f403648a6afb1d0866d87fbb/python_dateutil-2.8.0-py2.py3-
none-any.whl (https://files.pythonhosted.org/packages/41/17/c62faccbfbd163c7
f57f3844689e3a78bae1f403648a6afb1d0866d87fbb/python_dateutil-2.8.0-py2.py3-n
one-any.whl)
Collecting docutils<0.16,>=0.10 (from botocore<1.13.0,>=1.12.215->boto3->sma
rt-open>=1.7.0->gensim)
  Using cached https://files.pythonhosted.org/packages/22/cd/a6aa959dca61991
8ccb55023b4cb151949c64d4d5d55b3f4ffd7eee0c6e8/docutils-0.15.2-py3-none-any.w
hl (https://files.pythonhosted.org/packages/22/cd/a6aa959dca619918ccb55023b4
cb151949c64d4d5d55b3f4ffd7eee0c6e8/docutils-0.15.2-py3-none-any.whl)
Installing collected packages: numpy, scipy, six, boto, urllib3, jmespath, p
ython-dateutil, docutils, botocore, s3transfer, boto3, chardet, idna, certif
i, requests, smart-open, gensim
Successfully installed boto-2.49.0 boto3-1.9.215 botocore-1.12.215 certifi-2
019.6.16 chardet-3.0.4 docutils-0.15.2 gensim-3.8.0 idna-2.8 jmespath-0.9.4
 numpy-1.17.0 python-dateutil-2.8.0 requests-2.22.0 s3transfer-0.2.1 scipy-
1.3.1 six-1.12.0 smart-open-1.8.4 urllib3-1.25.3
```

In [2]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [3]:

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

In [4]:

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

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

In [5]:

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

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

Out[5]:

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

In [6]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]


project_data.head(2)
```

Out[6]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:

In [7]:

```
project_grade_category = []

for i in range(len(project_data)):
    a = project_data["project_grade_category"][i].replace(" ", "_")
    project_grade_category.append(a)

project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data["project_grade_category"] = project_grade_category
project_data.head(5)
```

Out[7]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

1.2 preprocessing of `project_subject_categories`

In [8]:

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

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

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [9]:

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

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

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

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

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

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

Clean Titles (Text preprocessing)

In [10]:

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

In [11]:

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

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

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

In [12]:

```
clean_titles = []

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

```
100%|██████████| 109248/109248 [00:03<00:00, 34499.61it/s]
```

In [13]:

```
project_data["clean_titles"] = clean_titles
```

In [14]:

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

Feature "Number of Words in Title"

In [15]:

```
title_word_count = []
for a in project_data["clean_titles"] :
    b = len(a.split())
    title_word_count.append(b)

project_data["title_word_count"] = title_word_count
project_data.head(5)
```

Out[15]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

1.3 Text preprocessing

In [16]:

```
# 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 [17]:

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

Out[17]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:

Clean Essays (Text preprocessing)

In [18]:

```
clean_essay = []

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

```
100%|██████████| 109248/109248 [01:13<00:00, 1487.44it/s]
```

In [19]:

```
project_data["clean_essays"] = clean_essay
```

In [20]:

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

Number of Words in Essay

In [21]:

```
essay_word_count = []
for ess in project_data["clean_essays"] :
    c = len(ess.split())
    essay_word_count.append(c)

project_data["essay_word_count"] = essay_word_count

project_data.head(5)
```

Out[21]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00:
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	00:
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	00:
473	100660	p234804	cbc0e38f522143b86d372f8b43d4cff3	Mrs.	GA	00:
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	01:

Train test Split

In [22]:

```
# train test split

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data, project_data['project_is_
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=
```

In [23]:

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

```
i fortunate enough use fairy tale stem kits classroom well stem journals stu
dents really enjoyed i would love implement lakeshore stem kits classroom ne
xt school year provide excellent engaging stem lessons my students come vari
ety backgrounds including language socioeconomic status many not lot experie
nce science engineering kits give materials provide exciting opportunities s
tudents each month i try several science stem steam projects i would use kit
s robot help guide science instruction engaging meaningful ways i adapt kits
current language arts pacing guide already teach material kits like tall tal
es paul bunyan johnny appleseed the following units taught next school year
i implement kits magnets motion sink vs float robots i often get units not k
now if i teaching right way using right materials the kits give additional i
deas strategies lessons prepare students science it challenging develop high
quality science activities these kits give materials i need provide students
science activities go along curriculum classroom although i things like magn
ets classroom i not know use effectively the kits provide right amount mater
ials show use appropriate way
==================================================
i teach high school english students learning behavioral disabilities my stu
dents vary ability level however ultimate goal increase students literacy le
vels this includes reading writing communication levels i teach really dynam
ic group students however students face lot challenges my students live pove
rty dangerous neighborhood despite challenges i students desire defeat chall
enges my students learning disabilities currently performing grade level my
students visual learners benefit classroom fulfills preferred learning style
the materials i requesting allow students prepared classroom necessary suppl
ies too often i challenged students come school unprepared class due economi
c challenges i want students able focus learning not able get school supplie
s the supplies last year students able complete written assignments maintain
classroom journal the chart paper used make learning visual class create pos
ters aid students learning the students access classroom printer the toner u
sed print student work completed classroom chromebooks i want try remove bar
riers students learning create opportunities learning one biggest barriers s
tudents not resources get pens paper folders my students able increase liter
acy skills project
==================================================
life moves pretty fast if not stop look around awhile could miss movie ferri
s bueller day off think back remember grandparents how amazing would able fl
ip book see day lives my second graders voracious readers they love read fic
tion nonfiction books their favorite characters include pete cat fly guy pig
gie elephant mercy watson they also love read insects space plants my studen
ts hungry bookworms my students eager learn read world around my kids love s
chool like little sponges absorbing everything around their parents work lon
g hours usually not see children my students usually cared grandparents fami
ly friend most students not someone speaks english home thus difficult stude
nts acquire language now think forward would not mean lot kids nieces nephew
s grandchildren able see day life today 30 years memories precious us able s
hare memories future generations rewarding experience as part social studies
```

```
curriculum students learning changes time students studying photos learn com
munity changed time in particular look photos study land buildings clothing 
schools changed time as culminating activity students capture slice history 
preserve scrap booking key important events young lives documented date loca
tion names students using photos home school create second grade memories th
eir scrap books preserve unique stories future generations enjoy your donati
on project provide second graders opportunity learn social studies fun creat
ive manner through scrapbooks children share story others historical documen
t rest lives
==================================================
a person person no matter small dr seuss i teach smallest students biggest e
nthusiasm learning my students learn many different ways using senses multip
le intelligences i use wide range techniques help students succeed students 
class come variety different backgrounds makes wonderful sharing experiences 
cultures including native americans our school caring community successful l
earners seen collaborative student project based learning classroom kinderga
rteners class love work hands materials many different opportunities practic
e skill mastered having social skills work cooperatively friends crucial asp
ect kindergarten curriculum montana perfect place learn agriculture nutritio
n my students love role play pretend kitchen early childhood classroom i sev
eral kids ask can try cooking real food i take idea create common core cooki
ng lessons learn important math writing concepts cooking delicious healthy f
ood snack time my students grounded appreciation work went making food knowl
edge ingredients came well healthy bodies this project would expand learning 
nutrition agricultural cooking recipes us peel apples make homemade applesau
ce make bread mix healthy plants classroom garden spring we also create cook
books printed shared families students gain math literature skills well life
long enjoyment healthy cooking nannan
==================================================
my classroom consists twenty two amazing sixth graders different cultures ba
ckgrounds they social bunch enjoy working partners working groups they hard 
working eager head middle school next year my job get ready make transition 
make smooth possible in order students need come school every day feel safe 
ready learn because getting ready head middle school i give lots choice choi
ce sit work order complete assignments choice projects etc part students fee
ling safe ability come welcoming encouraging environment my room colorful at
mosphere casual i want take ownership classroom all share together because t
ime limited i want ensure get time enjoy best abilities currently twenty two 
desks differing sizes yet desks similar ones students use middle school we a
lso kidney table crates seating i allow students choose spots working indepe
ndently groups more often not move desks onto crates believe not proven succ
essful making stay desks it i looking toward flexible seating option classro
om the students look forward work time move around room i would like get rid 
constricting desks move toward fun seating options i requesting various seat
ing students options sit currently i stool papasan chair i inherited previou
s sixth grade teacher well five milk crate seats i made i would like give op
tions reduce competition good seats i also requesting two rugs not seating o
ptions make classroom welcoming appealing in order students able write compl
ete work without desks i requesting class set clipboards finally due curricu
lum requires groups work together i requesting tables fold not using leave r
oom flexible seating options i know seating options much excited coming scho
ol thank support making classroom one students remember forever nannan
==================================================
```

In [24]:

```
# 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 [25]:

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

```
a person person no matter small dr seuss i teach smallest students biggest e
nthusiasm learning my students learn many different ways using senses multip
le intelligences i use wide range techniques help students succeed students 
class come variety different backgrounds makes wonderful sharing experiences
cultures including native americans our school caring community successful l
earners seen collaborative student project based learning classroom kinderga
rteners class love work hands materials many different opportunities practic
e skill mastered having social skills work cooperatively friends crucial asp
ect kindergarten curriculum montana perfect place learn agriculture nutritio
n my students love role play pretend kitchen early childhood classroom i sev
eral kids ask can try cooking real food i take idea create common core cooki
ng lessons learn important math writing concepts cooking delicious healthy f
ood snack time my students grounded appreciation work went making food knowl
edge ingredients came well healthy bodies this project would expand learning
nutrition agricultural cooking recipes us peel apples make homemade applesau
ce make bread mix healthy plants classroom garden spring we also create cook
books printed shared families students gain math literature skills well life
long enjoyment healthy cooking nannan
==================================================
```

In [26]:

```
# \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)
```

```
a person person no matter small dr seuss i teach smallest students biggest e
nthusiasm learning my students learn many different ways using senses multip
le intelligences i use wide range techniques help students succeed students 
class come variety different backgrounds makes wonderful sharing experiences
cultures including native americans our school caring community successful l
earners seen collaborative student project based learning classroom kinderga
rteners class love work hands materials many different opportunities practic
e skill mastered having social skills work cooperatively friends crucial asp
ect kindergarten curriculum montana perfect place learn agriculture nutritio
n my students love role play pretend kitchen early childhood classroom i sev
eral kids ask can try cooking real food i take idea create common core cooki
ng lessons learn important math writing concepts cooking delicious healthy f
ood snack time my students grounded appreciation work went making food knowl
edge ingredients came well healthy bodies this project would expand learning
nutrition agricultural cooking recipes us peel apples make homemade applesau
ce make bread mix healthy plants classroom garden spring we also create cook
books printed shared families students gain math literature skills well life
long enjoyment healthy cooking nannan
```

In [27]:

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

```
a person person no matter small dr seuss i teach smallest students biggest e
nthusiasm learning my students learn many different ways using senses multip
le intelligences i use wide range techniques help students succeed students 
class come variety different backgrounds makes wonderful sharing experiences
cultures including native americans our school caring community successful l
earners seen collaborative student project based learning classroom kinderga
rteners class love work hands materials many different opportunities practic
e skill mastered having social skills work cooperatively friends crucial asp
ect kindergarten curriculum montana perfect place learn agriculture nutritio
n my students love role play pretend kitchen early childhood classroom i sev
eral kids ask can try cooking real food i take idea create common core cooki
ng lessons learn important math writing concepts cooking delicious healthy f
ood snack time my students grounded appreciation work went making food knowl
edge ingredients came well healthy bodies this project would expand learning
nutrition agricultural cooking recipes us peel apples make homemade applesau
ce make bread mix healthy plants classroom garden spring we also create cook
books printed shared families students gain math literature skills well life
long enjoyment healthy cooking nannan
```

In [28]:

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

Preprocessed Train data (Essay)

In [29]:

```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays_train = []
# tqdm is for printing the status bar
for sentance in tqdm(X_train['clean_essays'].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.lower() not in stopwords)
    preprocessed_essays_train.append(sent.lower().strip())
```

```
100%|██████████| 49041/49041 [00:23<00:00, 2055.75it/s]
```

In [30]:

```
# after preprocesing
preprocessed_essays_train[20000]
```

Out[30]:

```
'students range academic levels honors students 95th percentile struggling s
tudents 1st percentile teach 50 boys 25 girls conduct learning style invento
ry students beginning year find learn best identify individual learning styl
e seat groups varied learning styles setting good examples others good role 
model student expectation classroom school focuses giving students leadershi
p roles classroom within school leadership roles given qualified student not
outstanding student everyone job completing work accurately behavior warrant
s position course let struggling student feel accomplished give job know par
t classroom community much kid makes often times incentive getting classroom
job brings best performance everyone school wide literacy goal every child r
ead one million words april 2017 kids easy goal reach simply matter get one 
month two kids may well told needed swim across atlantic inconceivable somew
here along way either never pushed read given exposure reading material perh
aps never priority home whatever reason truly bane existence goal classroom 
teacher help child find inner reader least inner listener audio books many d
ifferent reasons children struggle read hate read job classroom teacher help
succeed find way good book order find way reach reaching students getting in
terested books look differently students natural born readers simply need ma
terial keep reading however no longer live world handing student book enough
read children learn differently realize students need read books kids discus
s book move chapters children need technical aspect remain engaged like audi
o books research know audio books offer much originally thought important so
mething offer everyone nannan'
```

Preprocessed Test data (Essay)

In [31]:

```
preprocessed_essays_test = []
# tqdm is for printing the status bar
for sentence in tqdm(X_test['clean_essays'].values):
    sent = decontracted(sentence)
    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.lower() not in stopwords)
    preprocessed_essays_test.append(sent.lower().strip())
```

```
100%|██████████| 36052/36052 [00:17<00:00, 2057.81it/s]
```

In [32]:

```
preprocessed_essays_test[0]
```

Out[32]:

```
'special needs teacher work many students grades 3 5 included general educat
ion classroom students varying disabilities work students autism learning di
sabilities students love learn sometimes need different approach understand
concepts retain important skills taught class often need hands materials mak
e learning engaging meaningful see tangible representations learning student
s work hard day day learn fundamentals reading need exposure different mater
ials order cement understanding well fun way outlet learning using reading g
ames reinforcing skills already learning class fun interactive way students
relate not students learning playing also able practice social skills comple
ting tasks involving turn taking problem solving times conflict resolution n
annan'
```

Preprocessed Cross Validation data (essay)

In [33]:

```
preprocessed_essays_cv = []
# tqdm is for printing the status bar
for sentence in tqdm(X_cv['clean_essays'].values):
    sent = decontracted(sentence)
    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.lower() not in stopwords)
    preprocessed_essays_cv.append(sent.lower().strip())
```

```
100%|██████████| 24155/24155 [00:11<00:00, 2050.99it/s]
```

In [34]:

```
preprocessed_essays_cv[0]
```

Out[34]:

```
'quote try live change wish see world try relay message kiddos constantly pr
acticing growth mindset struggles persevere students student heart every mor
ning read school contract reminder follow code conduct work together heterog
eneous partnerships groups day focus bunch time collaborative conversations
academic area keeping task building language positive take active stance lea
rning class finds calming art time hope try integrate curriculum paintbrushe
s water colors would also like class use art projects gifts take home learn
basics shading water colors also growing artists items improve classroom giv
e students may not feel successful academic area feel understand area geomet
ry fractions painting build equity students representing math social studies
science vehicle visual arts students able express art'
```

1.4 Preprocessing of `project_title`

Preprocessing of Project Title for Train data

In [35]:

```
# similarly you can preprocess the titles also
preprocessed_titles_train = []

for titles in tqdm(X_train["clean_titles"]):
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_titles_train.append(title.lower().strip())
```

```
100%|██████████| 49041/49041 [00:01<00:00, 35957.45it/s]
```

In [36]:

```
preprocessed_titles_train[0]
```

Out[36]:

```
'no place like chrome'
```

Preprocessing of Project Title for Test data

In [37]:

```
preprocessed_titles_test = []

for titles in tqdm(X_test["clean_titles"]):
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_titles_test.append(title.lower().strip())
```

```
100%|██████████| 36052/36052 [00:01<00:00, 35537.63it/s]
```

In [38]:

```
preprocessed_titles_test[0]
```

Out[38]:

```
'fun games reading'
```

Preprocessing of Project Title for CV data

In [39]:

```
preprocessed_titles_cv = []

for titles in tqdm(X_cv["clean_titles"]):
    title = decontracted(titles)
    title = title.replace('\\r', ' ')
    title = title.replace('\\"', ' ')
    title = title.replace('\\n', ' ')
    title = re.sub('[^A-Za-z0-9]+', ' ', title)
    title = ' '.join(f for f in title.split() if f not in stopwords)
    preprocessed_titles_cv.append(title.lower().strip())
```

```
100%|██████████| 24155/24155 [00:00<00:00, 35233.97it/s]
```

In [40]:

```
preprocessed_titles_cv[0]
```

Out[40]:

```
'visual art integration 04 28 16'
```

1.5 Preparing data for models

In [41]:

```
project_data.columns
```

Out[41]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'project_grade_category', 'clean_categories', 'clean_subcategories',
       'clean_titles', 'title_word_count', 'clean_essays', 'essay_word_coun
t'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

In [42]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer

vectorizer_proj = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
vectorizer_proj.fit(X_train['clean_categories'].values)

categories_one_hot_train = vectorizer_proj.transform(X_train['clean_categories'].values)
categories_one_hot_test = vectorizer_proj.transform(X_test['clean_categories'].values)
categories_one_hot_cv = vectorizer_proj.transform(X_cv['clean_categories'].values)

print(vectorizer_proj.get_feature_names())


print("Shape of matrix of Train data after one hot encoding ",categories_one_hot_train.shap
print("Shape of matrix of Test data after one hot encoding ",categories_one_hot_test.shape)
print("Shape of matrix of CV data after one hot encoding ",categories_one_hot_cv.shape)
```

```
['Care_Hunger', 'Math_Science', 'History_Civics', 'Music_Arts', 'Warmth', 'S
pecialNeeds', 'AppliedLearning', 'Literacy_Language', 'Health_Sports']
Shape of matrix of Train data after one hot encoding  (49041, 9)
Shape of matrix of Test data after one hot encoding  (36052, 9)
Shape of matrix of CV data after one hot encoding  (24155, 9)
```

In [43]:

```
# we use count vectorizer to convert the values into one
vectorizer_sub_proj = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercas
vectorizer_sub_proj.fit(X_train['clean_subcategories'].values)

sub_categories_one_hot_train = vectorizer_sub_proj.transform(X_train['clean_subcategories']
sub_categories_one_hot_test = vectorizer_sub_proj.transform(X_test['clean_subcategories'].v
sub_categories_one_hot_cv = vectorizer_sub_proj.transform(X_cv['clean_subcategories'].value


print(vectorizer_sub_proj.get_feature_names())



print("Shape of matrix of Train data after one hot encoding ",sub_categories_one_hot_train.
print("Shape of matrix of Test data after one hot encoding ",sub_categories_one_hot_test.sh
print("Shape of matrix of Cross Validation data after one hot encoding ",sub_categories_one
```

```
['Mathematics', 'SocialSciences', 'Gym_Fitness', 'FinancialLiteracy', 'Envir
onmentalScience', 'CommunityService', 'VisualArts', 'Warmth', 'History_Geogr
aphy', 'NutritionEducation', 'Health_Wellness', 'Extracurricular', 'Literatu
re_Writing', 'PerformingArts', 'ForeignLanguages', 'AppliedSciences', 'Early
Development', 'Health_LifeScience', 'College_CareerPrep', 'Music', 'Civics_G
overnment', 'TeamSports', 'Economics', 'Literacy', 'Care_Hunger', 'ESL', 'Ot
her', 'SpecialNeeds', 'CharacterEducation', 'ParentInvolvement']
Shape of matrix of Train data after one hot encoding  (49041, 30)
Shape of matrix of Test data after one hot encoding  (36052, 30)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 30)
```

In [44]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
my_counter = Counter()
for state in project_data['school_state'].values:
    my_counter.update(state.split())
```

In [45]:

```
school_state_cat_dict = dict(my_counter)
sorted_school_state_cat_dict = dict(sorted(school_state_cat_dict.items(), key=lambda kv: kv
```

In [46]:

```
## we use count vectorizer to convert the values into one hot encoded features

vectorizer_states = CountVectorizer(vocabulary=list(sorted_school_state_cat_dict.keys()), l
vectorizer_states.fit(X_train['school_state'].values)

school_state_categories_one_hot_train = vectorizer_states.transform(X_train['school_state']
school_state_categories_one_hot_test = vectorizer_states.transform(X_test['school_state'].v
school_state_categories_one_hot_cv = vectorizer_states.transform(X_cv['school_state'].value

print(vectorizer_states.get_feature_names())



print("Shape of matrix of Train data after one hot encoding ",school_state_categories_one_h
print("Shape of matrix of Test data after one hot encoding ",school_state_categories_one_hc
print("Shape of matrix of Cross Validation data after one hot encoding ",school_state_categ
```

```
['ID', 'NY', 'WA', 'RI', 'NM', 'MI', 'HI', 'IA', 'AR', 'NH', 'WY', 'NE', 'N
C', 'KY', 'UT', 'OK', 'IN', 'CO', 'AZ', 'FL', 'WV', 'SD', 'GA', 'WI', 'CA',
'SC', 'KS', 'DC', 'MO', 'DE', 'VT', 'MS', 'AL', 'NV', 'ME', 'OR', 'TX', 'M
A', 'CT', 'NJ', 'MT', 'MD', 'MN', 'TN', 'OH', 'PA', 'IL', 'VA', 'ND', 'LA',
'AK']
Shape of matrix of Train data after one hot encoding  (49041, 51)
Shape of matrix of Test data after one hot encoding  (36052, 51)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 51)
```

In [47]:

```
my_counter = Counter()
for project_grade in project_data['project_grade_category'].values:
    my_counter.update(project_grade.split())
```

In [48]:

```
project_grade_cat_dict = dict(my_counter)
sorted_project_grade_cat_dict = dict(sorted(project_grade_cat_dict.items(), key=lambda kv:
```

In [49]:

```
## we use count vectorizer to convert the values into one hot encoded features

vectorizer_grade = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()), l
vectorizer_grade.fit(X_train['project_grade_category'].values)

project_grade_categories_one_hot_train = vectorizer_grade.transform(X_train['project_grade_
project_grade_categories_one_hot_test = vectorizer_grade.transform(X_test['project_grade_ca
project_grade_categories_one_hot_cv = vectorizer_grade.transform(X_cv['project_grade_catego

print(vectorizer_grade.get_feature_names())



print("Shape of matrix of Train data after one hot encoding ",project_grade_categories_one_
print("Shape of matrix of Test data after one hot encoding ",project_grade_categories_one_h
print("Shape of matrix of Cross Validation data after one hot encoding ",project_grade_cate
```

```
['Grades_PreK-2', 'Grades_6-8', 'Grades_9-12', 'Grades_3-5']
Shape of matrix of Train data after one hot encoding  (49041, 4)
Shape of matrix of Test data after one hot encoding  (36052, 4)
Shape of matrix of Cross Validation data after one hot encoding  (24155, 4)
```

In [50]:

```
my_counter = Counter()
for teacher_prefix in project_data['teacher_prefix'].values:
    teacher_prefix = str(teacher_prefix)
    my_counter.update(teacher_prefix.split())
```

In [51]:

```
teacher_prefix_cat_dict = dict(my_counter)
sorted_teacher_prefix_cat_dict = dict(sorted(teacher_prefix_cat_dict.items(), key=lambda kv
```

In [52]:

```
## we use count vectorizer to convert the values into one hot encoded features
## Unlike the previous Categories this category returns a
## ValueError: np.nan is an invalid document, expected byte or unicode string.
## The link below explains h0w to tackle such discrepancies.
## https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn-valueerror-

vectorizer_teacher = CountVectorizer(vocabulary=list(sorted_teacher_prefix_cat_dict.keys())
vectorizer_teacher.fit(X_train['teacher_prefix'].values.astype("U"))

teacher_prefix_categories_one_hot_train = vectorizer_teacher.transform(X_train['teacher_pre
teacher_prefix_categories_one_hot_test = vectorizer_teacher.transform(X_test['teacher_prefi
teacher_prefix_categories_one_hot_cv = vectorizer_teacher.transform(X_cv['teacher_prefix'].

print(vectorizer_teacher.get_feature_names())


print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_train.sha
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_test.shap
print("Shape of matrix after one hot encoding ",teacher_prefix_categories_one_hot_cv.shape)
```

```
['nan', 'Dr.', 'Teacher', 'Ms.', 'Mrs.', 'Mr.']
Shape of matrix after one hot encoding  (49041, 6)
Shape of matrix after one hot encoding  (36052, 6)
Shape of matrix after one hot encoding  (24155, 6)
```

1.5.2 Vectorizing Text data

a) Bag of words Train Data (Essays)

In [53]:

```
# We are considering only the words which appeared in at least 10 documents(rows or project
vectorizer_bow_essay = CountVectorizer(min_df=10)

vectorizer_bow_essay.fit(preprocessed_essays_train)

text_bow_train = vectorizer_bow_essay.transform(preprocessed_essays_train)



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

```
Shape of matrix after one hot encoding  (49041, 12054)
```

b) Bag of words Test Data (Essays)

In [54]:

```
text_bow_test = vectorizer_bow_essay.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 12054)
```

c) Bag of words CV Data (Essays)

In [55]:

```
text_bow_cv = vectorizer_bow_essay.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encoding ",text_bow_cv.shape)
```

```
Shape of matrix after one hot encoding  (24155, 12054)
```

d) Bag of words train Data (Titles)

In [56]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_bow_title = CountVectorizer(min_df=10)



vectorizer_bow_title.fit(preprocessed_titles_train)



title_bow_train = vectorizer_bow_title.transform(preprocessed_titles_train)



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

```
Shape of matrix after one hot encoding  (49041, 1968)
```

e) Bag of words Test Data (Titles)

In [57]:

```
title_bow_test = vectorizer_bow_title.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encoding ",title_bow_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 1968)
```

f) Bag of words Data (Titles)

In [58]:

```
title_bow_cv = vectorizer_bow_title.transform(preprocessed_titles_cv)
print("Shape of matrix after one hot encoding ",title_bow_cv.shape)
```

```
Shape of matrix after one hot encoding  (24155, 1968)
```

1.5.2.2 TFIDF vectorizer

a) TFIDF vectorizer Train Data (Essays)

In [59]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(preprocessed_essays_train)

text_tfidf_train = vectorizer_tfidf_essay.transform(preprocessed_essays_train)



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

```
Shape of matrix after one hot encoding  (49041, 12054)
```

b) TFIDF vectorizer Test Data (Essays)

In [60]:

```
text_tfidf_test = vectorizer_tfidf_essay.transform(preprocessed_essays_test)
print("Shape of matrix after one hot encoding ",text_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 12054)
```

c) TFIDF vectorizer CV Data (Essays)

In [61]:

```
text_tfidf_cv = vectorizer_tfidf_essay.transform(preprocessed_essays_cv)
print("Shape of matrix after one hot encoding ",text_tfidf_cv.shape)
```

```
Shape of matrix after one hot encoding  (24155, 12054)
```

c) TFIDF vectorizer Train Data (Titles)

In [62]:

```
vectorizer_tfidf_titles = TfidfVectorizer(ngram_range=(2,2), min_df=10)

vectorizer_tfidf_titles.fit(preprocessed_titles_train)
title_tfidf_train = vectorizer_tfidf_titles.transform(preprocessed_titles_train)



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

```
Shape of matrix after one hot encoding  (49041, 1241)
```

d) TFIDF vectorizer Test Data (Titles)

In [63]:

```
title_tfidf_test = vectorizer_tfidf_titles.transform(preprocessed_titles_test)
print("Shape of matrix after one hot encoding ",title_tfidf_test.shape)
```

```
Shape of matrix after one hot encoding  (36052, 1241)
```

e) TFIDF vectorizer CV Data (Titles)

In [64]:

```
title_tfidf_cv = vectorizer_tfidf_titles.transform(preprocessed_titles_cv)
print("Shape of matrix after one hot encoding ",title_tfidf_cv.shape)
```

```
Shape of matrix after one hot encoding  (24155, 1241)
```

C) Using Pretrained Models : AVG W2V

In [65]:

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

model = loadGloveModel('glove.42B.300d.txt')


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

print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

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

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


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

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

```
Loading Glove Model

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-65-40504d584192> in <module>
     12     return model
     13
---> 14 model = loadGloveModel('glove.42B.300d.txt')
     15
     16

<ipython-input-65-40504d584192> in loadGloveModel(gloveFile)
      2 def loadGloveModel(gloveFile):
      3     print ("Loading Glove Model")
----> 4     f = open(gloveFile,'r', encoding="utf8")
      5     model = {}
      6     for line in tqdm(f):

FileNotFoundError: [Errno 2] No such file or directory: 'glove.42B.300d.txt'
```

In [66]:

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

Train Essay

In [67]:

```
# average Word2Vec
# compute average word2vec for each review.

avg_w2v_vectors_train = [];

for sentence in tqdm(X_train["clean_essays"]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_train.append(vector)

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

```
100%|██████████| 49041/49041 [00:16<00:00, 2900.81it/s]

49041
300
```

Test Essay

In [68]:

```
# average Word2Vec
# compute average word2vec for each review.

avg_w2v_vectors_test = [];

for sentence in tqdm(X_test["clean_essays"]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_test.append(vector)

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

```
100%|██████████| 36052/36052 [00:12<00:00, 2920.22it/s]

36052
300
```

Cross validation Essay

In [69]:

```
avg_w2v_vectors_cv = [];

for sentence in tqdm(X_cv["clean_essays"]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_cv.append(vector)

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

```
100%|██████████| 24155/24155 [00:08<00:00, 2891.66it/s]

24155
300
```

train Titles

In [70]:

```
avg_w2v_vectors_titles_train = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(X_train["clean_titles"]): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_train.append(vector)

print(len(avg_w2v_vectors_titles_train))
print(len(avg_w2v_vectors_titles_train[0]))
```

```
100%|██████████| 49041/49041 [00:00<00:00, 58925.56it/s]

49041
300
```

Test Titles

In [71]:

```
avg_w2v_vectors_titles_test = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(X_test["clean_titles"]): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_test.append(vector)

print(len(avg_w2v_vectors_titles_test))
print(len(avg_w2v_vectors_titles_test[0]))
```

```
100%|██████████| 36052/36052 [00:00<00:00, 57293.30it/s]

36052
300
```

CV Titles

In [72]:

```
avg_w2v_vectors_titles_cv = []; # the avg-w2v for each sentence/review is stored in this li
for sentence in tqdm(X_cv["clean_titles"]): # for each title
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_cv.append(vector)

print(len(avg_w2v_vectors_titles_cv))
print(len(avg_w2v_vectors_titles_cv[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 57484.91it/s]

24155
300
```

D) Using Pretrained Models: TFIDF weighted W2V

Train - Essays

In [73]:

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

In [74]:

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

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

```
100%|██████████| 49041/49041 [01:50<00:00, 445.75it/s]

49041
300
```

Test essays

In [75]:

```
# compute average word2vec for each review.

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

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

```
100%|██████████| 36052/36052 [01:19<00:00, 455.48it/s]

36052
300
```

CV essays

In [76]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv["clean_essays"]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettir
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_cv.append(vector)

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

```
100%|██████████| 24155/24155 [00:54<00:00, 447.06it/s]

24155
300
```

Train Titles

In [77]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train["clean_titles"])
# 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 [78]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_titles_train = [];

for sentence in tqdm(X_train["clean_titles"]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_train.append(vector)

print(len(tfidf_w2v_vectors_titles_train))
print(len(tfidf_w2v_vectors_titles_train[0]))
```

```
100%|██████████| 49041/49041 [00:01<00:00, 27796.09it/s]

49041
300
```

Test Titles

In [79]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_titles_test = [];

for sentence in tqdm(X_test["clean_titles"]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_test.append(vector)

print(len(tfidf_w2v_vectors_titles_test))
print(len(tfidf_w2v_vectors_titles_test[0]))
```

```
100%|██████████| 36052/36052 [00:01<00:00, 24941.13it/s]

36052
300
```

CV Titles

In [80]:

```
# compute average word2vec for each review.

tfidf_w2v_vectors_titles_cv = [];

for sentence in tqdm(X_cv["clean_titles"]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentenc
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # gettin
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_titles_cv.append(vector)

print(len(tfidf_w2v_vectors_titles_cv))
print(len(tfidf_w2v_vectors_titles_cv[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 26973.01it/s]

24155
300
```

1.5.3 Vectorizing Numerical features

a) Price

In [81]:

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

In [82]:

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

In [83]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikitlearn.org/stable/modules/generated/sklearn.preproc
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import Normalizer
from sklearn import preprocessing
price_scalar = MinMaxScaler()
price_scalar.fit(X_train['price'].values.reshape(-1,1)) # finding the mean and standarddevi
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
price_train = price_scalar.transform(X_train['price'].values.reshape(-1, 1))
price_train
# Now standardize the data with above maen and variance.
price_test = price_scalar.transform(X_test['price'].values.reshape(-1, 1))
price_test
# Now standardize the data with above maen and variance.
price_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_cv
```

Out[83]:

```
array([[0.00684016],
       [0.03753334],
       [0.00442175],
       ...,
       [0.01682884],
       [0.0035406 ],
       [0.02043045]])
```

In [84]:

```
print("After vectorizations")
print(price_train.shape, y_train.shape)
print(price_cv.shape, y_cv.shape)
print(price_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

b) Quantity

In [85]:

```
price_scalar.fit(X_train['quantity'].values.reshape(-1,1)) # finding the mean and standard
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
quantity_train = price_scalar.transform(X_train['quantity'].values.reshape(-1, 1))
quantity_train
# Now standardize the data with above maen and variance.
quantity_cv = price_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
quantity_cv
# Now standardize the data with above maen and variance.
quantity_test = price_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
quantity_test
```

Out[85]:

```
array([[0.00616333],
       [0.03389831],
       [0.        ],
       ...,
       [0.00924499],
       [0.01232666],
       [0.01078582]])
```

In [86]:

```
print("After vectorizations")
print(quantity_train.reshape, y_train.shape)
print(quantity_cv.shape, y_cv.shape)
print(quantity_test.shape, y_test.shape)
```

```
After vectorizations
<built-in method reshape of numpy.ndarray object at 0x7f5e14d770d0> (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

c) Number of Projects previously proposed by Teacher

In [87]:

```
price_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,
#print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0]
# Now standardize the data with above maen and variance.
prev_projects_train = price_scalar.transform(X_train['teacher_number_of_previously_posted_p
prev_projects_train
# Now standardize the data with above maen and variance.
prev_projects_cv = price_scalar.transform(X_cv['teacher_number_of_previously_posted_project
prev_projects_cv
# Now standardize the data with above maen and variance.
prev_projects_test = price_scalar.transform(X_test['teacher_number_of_previously_posted_pro
prev_projects_test
```

Out[87]:

```
array([[0.00221729],
       [0.        ],
       [0.11529933],
       ...,
       [0.09756098],
       [0.00665188],
       [0.00221729]])
```

In [88]:

```
print("After vectorizations")
print(prev_projects_train.shape, y_train.shape)
print(prev_projects_cv.shape, y_cv.shape)
print(prev_projects_test.shape, y_test.shape)
```

```
After vectorizations
(49041, 1) (49041,)
(24155, 1) (24155,)
(36052, 1) (36052,)
```

Assignment 7: Decision Trees

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

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

2. **Hyper paramter tuning (best `depth` in range [4,6, 8, 9,10,12,14,17] , and the best `min_samples_split` in range [2,10,20,30,40,50])**

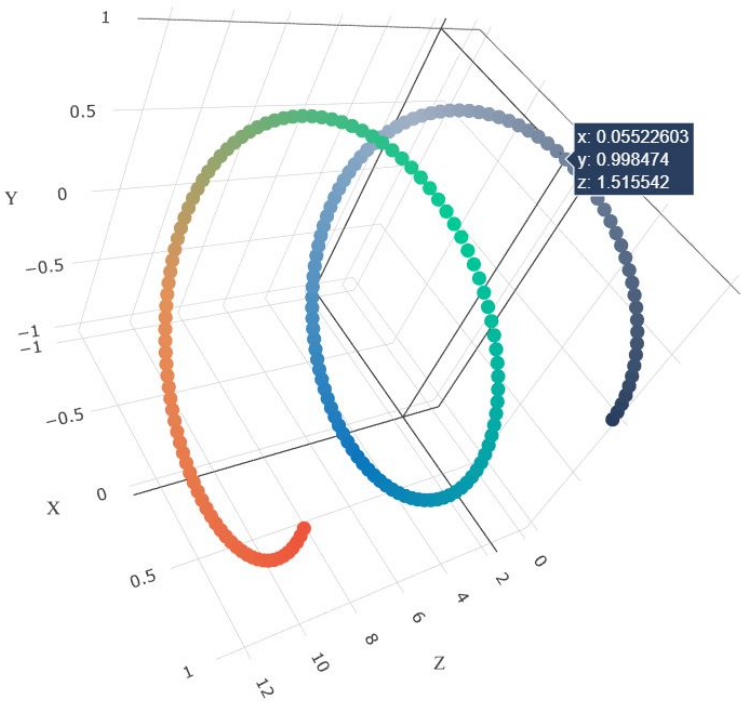
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data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

3. **Graphviz**

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

4. **Representation of results**

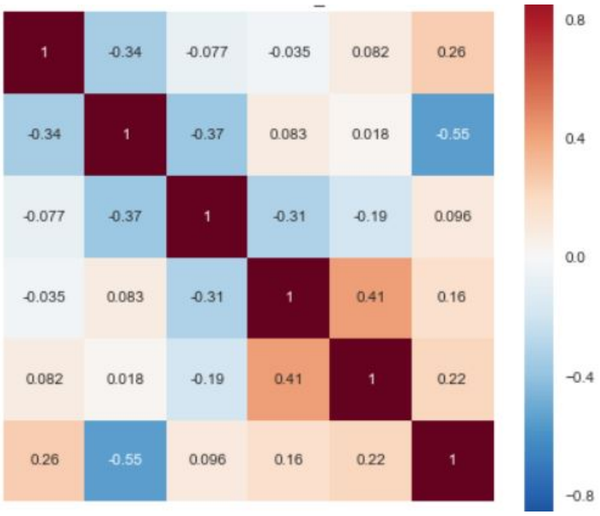
- You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure



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



seaborn heat maps (https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **min_sample_split**, 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 (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points
- Once after you plot the confusion matrix with the test data, get all the `false positive data points`
  - Plot the WordCloud WordCloud (https://www.geeksforgeeks.org/generating-word-cloud-python/)
  - Plot the box plot with the `price` of these `false positive data points`
  - Plot the pdf with the `teacher_number_of_previously_posted_projects` of these `false positive data points`

5. **[Task-2]**

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

6. **Conclusion**

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

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakage, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.
4. For more details please go through this link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2.1 Set 1: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

In [89]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,school_state_categori
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,school_state_categories
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,school_state_categories_one
```

In [90]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 14125) (49041,)
(24155, 14125) (24155,)
(36052, 14125) (36052,)
============================================================================
========================
```

In [91]:

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

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

    return y_data_pred
```

A) Gridsearch-cv

In [92]:

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

In [93]:

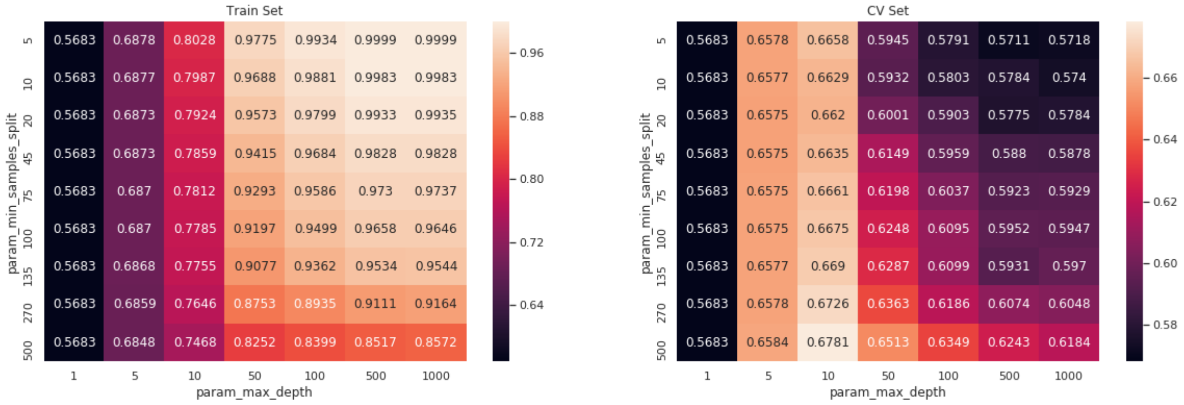
```
dt1 = DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10, 20,

clf1 = GridSearchCV(dt1, parameters, cv= 2, scoring='roc_auc',n_jobs=-1,return_train_score=
clf1.fit(X_tr, y_train)
```

Out[93]:

```
GridSearchCV(cv=2, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight='balanced',
                                              criterion='gini', max_depth=No
ne,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort=False, random_state=No
ne,
                                              splitter='best'),
             iid='warn', n_jobs=-1,
             param_grid={'max_depth': [1, 5, 10, 50, 100, 500, 1000],
                         'min_samples_split': [5, 10, 20, 45, 75, 100, 135,
270,
                                               500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [94]:

```
import seaborn as sns; sns.set()


max_scores1 = pd.DataFrame(clf1.cv_results_).groupby(['param_min_samples_split', 'param_max


fig, ax = plt.subplots(1,2, figsize=(20,6))

sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])

ax[0].set_title('Train Set')
ax[1].set_title('CV Set')

plt.show()
```



Train Set

param_min_samples_split \ param_max_depth	1	5	10	50	100	500	1000
5	0.5683	0.6878	0.8028	0.9775	0.9934	0.9999	0.9999
10	0.5683	0.6877	0.7987	0.9688	0.9881	0.9983	0.9983
20	0.5683	0.6873	0.7924	0.9573	0.9799	0.9933	0.9935
45	0.5683	0.6873	0.7859	0.9415	0.9684	0.9828	0.9828
75	0.5683	0.687	0.7812	0.9293	0.9586	0.973	0.9737
100	0.5683	0.687	0.7785	0.9197	0.9499	0.9658	0.9646
135	0.5683	0.6868	0.7755	0.9077	0.9362	0.9534	0.9544
270	0.5683	0.6859	0.7646	0.8753	0.8935	0.9111	0.9164
500	0.5683	0.6848	0.7468	0.8252	0.8399	0.8517	0.8572

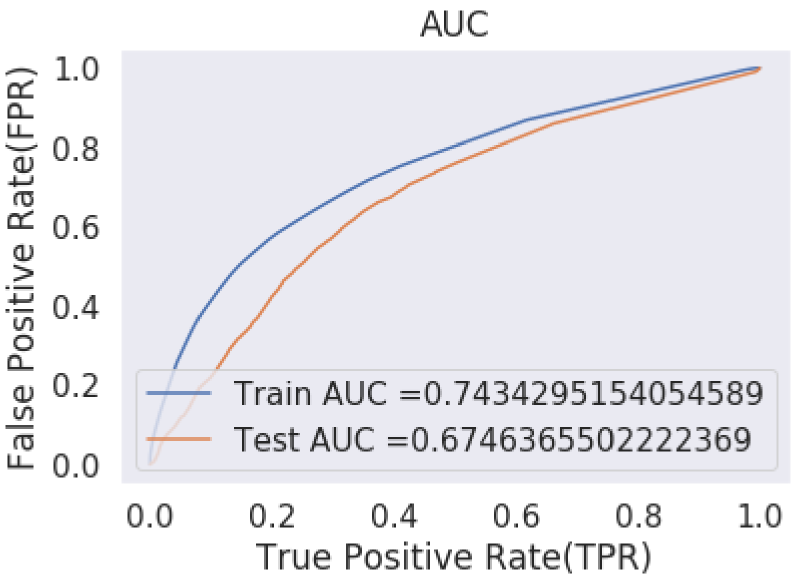
CV Set

param_min_samples_split \ param_max_depth	1	5	10	50	100	500	1000
5	0.5683	0.6578	0.6658	0.5945	0.5791	0.5711	0.5718
10	0.5683	0.6577	0.6629	0.5932	0.5803	0.5784	0.574
20	0.5683	0.6575	0.662	0.6001	0.5903	0.5775	0.5784
45	0.5683	0.6575	0.6635	0.6149	0.5959	0.588	0.5878
75	0.5683	0.6575	0.6661	0.6198	0.6037	0.5923	0.5929
100	0.5683	0.6575	0.6675	0.6248	0.6095	0.5952	0.5947
135	0.5683	0.6577	0.669	0.6287	0.6099	0.5931	0.597
270	0.5683	0.6578	0.6726	0.6363	0.6186	0.6074	0.6048
500	0.5683	0.6584	0.6781	0.6513	0.6349	0.6243	0.6184

B) Train model using the best hyper-parameter value

In [107]:

```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.m
from sklearn.metrics import roc_curve, auc
model = DecisionTreeClassifier(max_depth = 10, min_samples_split = 500,class_weight = 'bala
model.fit(X_tr, y_train)
clfV1=DecisionTreeClassifier (class_weight = 'balanced',max_depth=3,min_samples_split=500)
# for visulation
clfV1.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr)
y_test_pred = batch_predict(model, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



D) Confusion Matrix

In [108]:

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

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

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

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t
    predictions = []
    global predictions1# make it global
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    predictions1= predictions
    return predictions
```

Train Data

In [109]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2475040382279047 for threshold 0.36
[[ 4084  3342]
 [ 9334 32281]]
```

In [110]:

```
conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

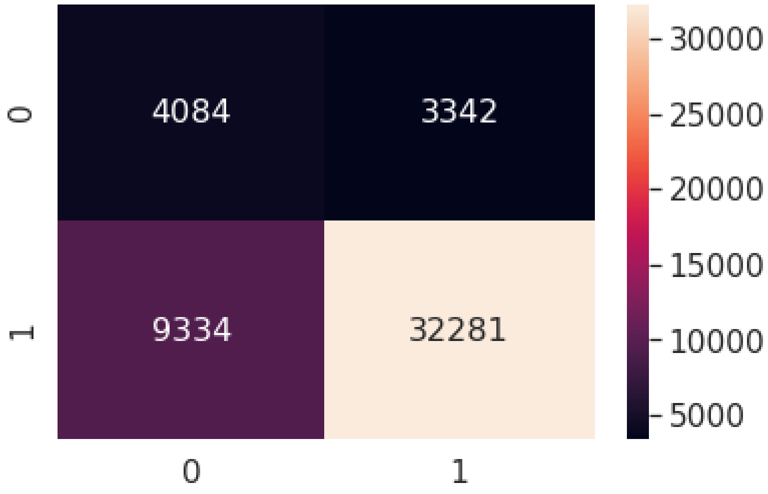
```
the maximum value of tpr*(1-fpr) 0.2475040382279047 for threshold 0.36
```

In [111]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[111]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e140872b0>
```



Test Data

In [112]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999294479210055 for threshold 0.36
[[ 2715  2744]
 [ 7295 23298]]
```

In [113]:

```
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_threshc
```

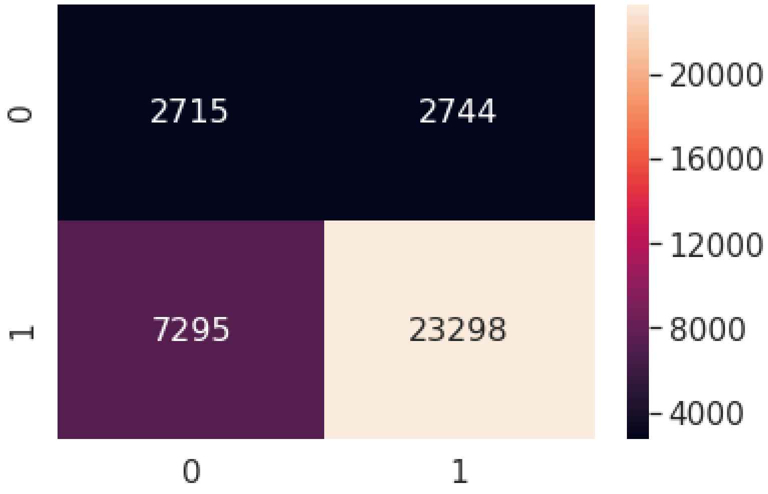
```
the maximum value of tpr*(1-fpr) 0.24999294479210055 for threshold 0.36
```

In [114]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[114]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e13e20390>
```



Feature aggregation

In [123]:

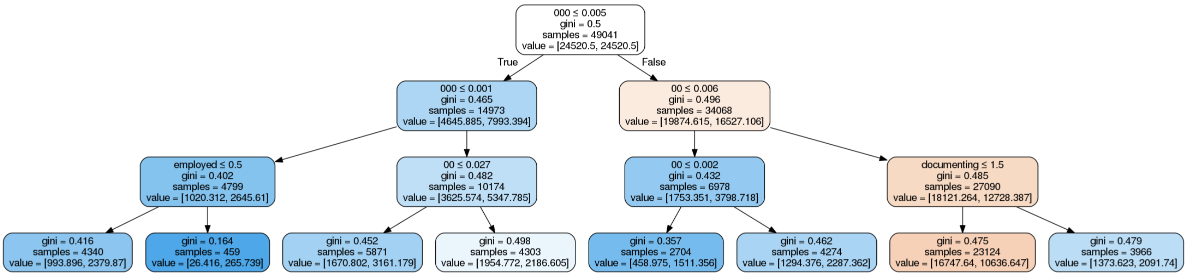
```
f1=vectorizer_proj.get_feature_names()
f2=vectorizer_sub_proj.get_feature_names()
f3= vectorizer_states.get_feature_names()
f4= vectorizer_grade.get_feature_names()
f5 = vectorizer_teacher.get_feature_names()
fbow= vectorizer_bow_essay.get_feature_names()
ftbow = vectorizer_bow_title.get_feature_names()
ftfidf = vectorizer_tfidf_essay.get_feature_names()
fttfidft = vectorizer_tfidf_titles.get_feature_names()

feature_agg_bow = f1 + f2 + f3 + f4 + f5 + fbow + ftbow
feature_agg_tfidf = f1 + f2 + f3 + f4 + f5 + ftfidf + fttfidft
# p is price, q is quantity, t is teacher previous year projects
feature_agg_bow.append('price')
feature_agg_tfidf.append('price')
feature_agg_bow.append('quantity')
feature_agg_tfidf.append('quantity')
feature_agg_bow.append('teacher_previous_projects')
feature_agg_tfidf.append('teacher_previous_projects')
```

Visualizing Decision Tree

In [124]:

```
import warnings
warnings.filterwarnings("ignore")
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus

dot_data = StringIO()
export_graphviz(clfV1, out_file=dot_data, filled=True, rounded=True, special_characters=Tru
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[124]:



False Positives

In [125]:

```
fpi = []

for i in range(len(y_test)) :
    if (y_test.values[i] == 0) & (predictions1[i] == 1) :
        fpi.append(i)
```

In [126]:

```
fp_essay1 = []
for i in fpi :
    fp_essay1.append(X_test['clean_essays'].values[i])
```

In [127]:

```
len(fp_essay1)
```

Out[127]:

```
2744
```

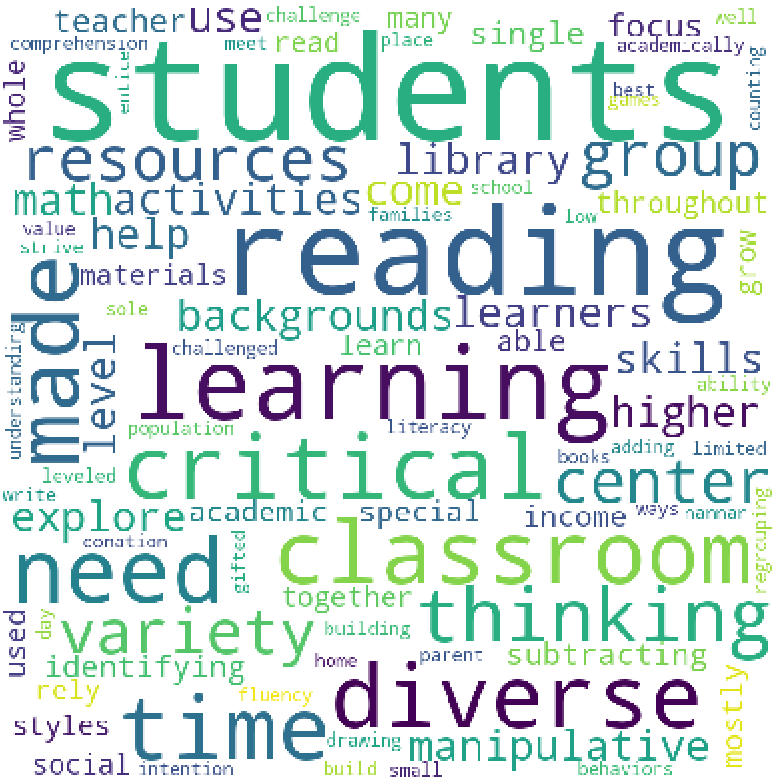
In [ ]:

```

```

Word Cloud (False positives essay)

In [128]:

```
from wordcloud import WordCloud, STOPWORDS

comment_words = ' '
stopwords = set(STOPWORDS)

for val in fp_essay1 :
    val = str(val)
    tokens = val.split()
for i in range(len(tokens)):
    tokens[i] = tokens[i].lower()
for words in tokens :
    comment_words = comment_words + words + ' '

wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords = stc


plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



Building DataFrame of False Positives

In [129]:

```
cols = X_test.columns
X_test_falsePos1 = pd.DataFrame(columns=cols)
```

In [130]:

```
for i in fpi :
    X_test_falsePos1 = X_test_falsePos1.append(X_test.filter(items=[i], axis=0))
```

In [131]:

```
len(X_test_falsePos1)
```

Out[131]:

```
2744
```

In [132]:

```
X_test_falsePos1.head(2)
```

Out[132]:

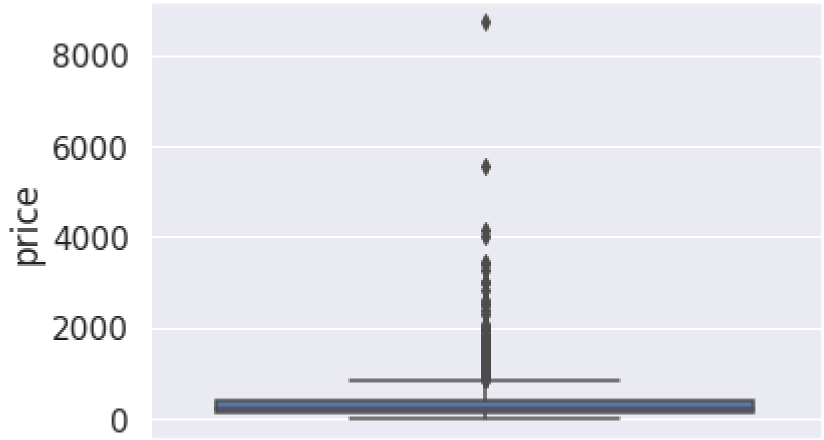
	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Dat
35	136188	p166914	dd32fbe60f0d5981c9f8822db9bfb80e	Ms.	NY	2016 10-0 13:53:2
65	77315	p007303	25cf951f507a0aaa6f251bbc02ab45a7	Mrs.	NC	2016 08-0 00:00:5

2 rows × 22 columns

box plot for Price

In [133]:

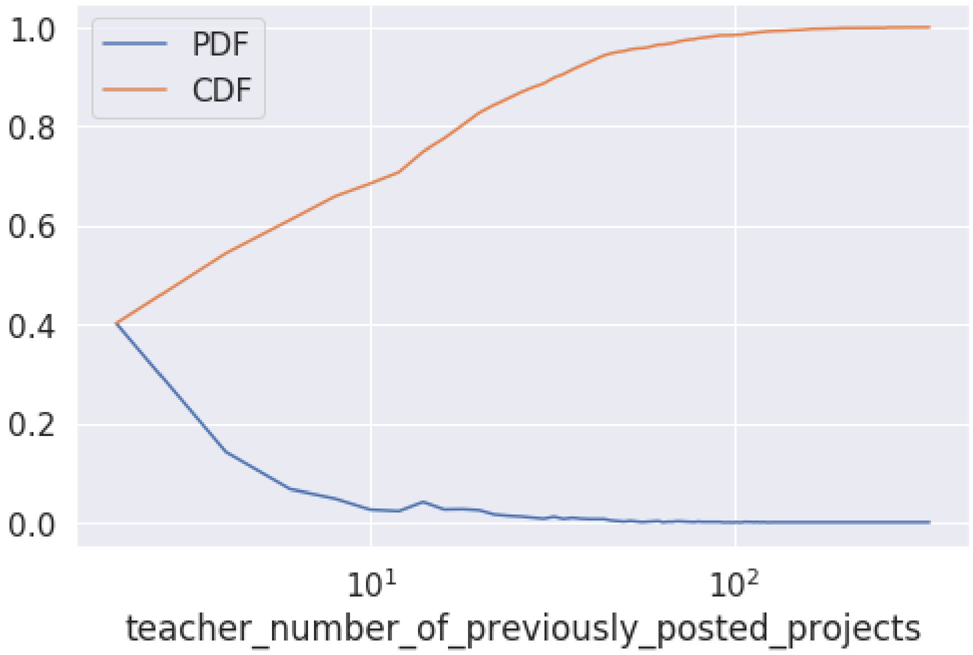
```
graph = sns.boxplot(y='price', data=X_test_falsePos1)
```



PDF on teacher_number_of_previously_posted_projects

In [134]:

```
plt.figure(figsize=(8,5))

counts, bin_edges = np.histogram(X_test_falsePos1['teacher_number_of_previously_posted_proj
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
pdfP, = plt.plot(bin_edges[1:], pdf)
cdfP, = plt.plot(bin_edges[1:], cdf)
plt.legend([pdfP, cdfP], ["PDF", "CDF"])
plt.xscale('log')
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



Set 2 : categorical, numerical features + project_title(TFIDF) + preprocessed_essay (TFIDF)

In [135]:

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

X_tr2 = hstack((categories_one_hot_train, sub_categories_one_hot_train,school_state_categor
X_te2 = hstack((categories_one_hot_test, sub_categories_one_hot_test,school_state_categorie
X_cr2 = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,school_state_categories_or
```

In [136]:

```
print("Final Data matrix")
print(X_tr2.shape, y_train.shape)
print(X_cr2.shape, y_cv.shape)
print(X_te2.shape, y_test.shape)
```

```
Final Data matrix
(49041, 13398) (49041,)
(24155, 13398) (24155,)
(36052, 13398) (36052,)
```

GridSearch CV

In [137]:

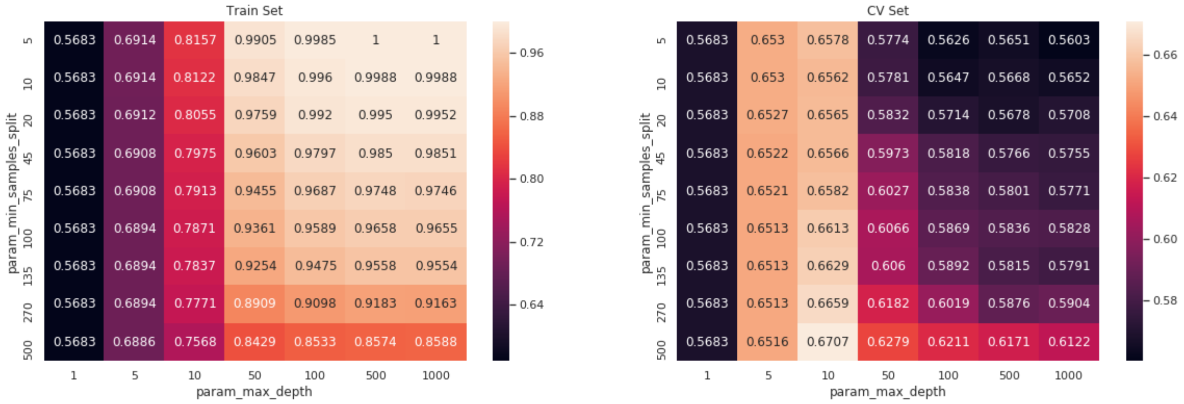
```
dt2 = DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10, 20,

clf2 = GridSearchCV(dt2, parameters, cv= 2, scoring='roc_auc',n_jobs=-1,return_train_score=
clf2.fit(X_tr2, y_train)
```

Out[137]:

```
GridSearchCV(cv=2, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight='balanced',
                                              criterion='gini', max_depth=No
ne,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort=False, random_state=No
ne,
                                              splitter='best'),
             iid='warn', n_jobs=-1,
             param_grid={'max_depth': [1, 5, 10, 50, 100, 500, 1000],
                         'min_samples_split': [5, 10, 20, 45, 75, 100, 135,
270,
                                               500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [138]:

```
import seaborn as sns; sns.set()


max_scores2 = pd.DataFrame(clf2.cv_results_).groupby(['param_min_samples_split', 'param_max


fig, ax = plt.subplots(1,2, figsize=(20,6))

sns.heatmap(max_scores2.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores2.mean_test_score, annot = True, fmt='.4g', ax=ax[1])

ax[0].set_title('Train Set')
ax[1].set_title('CV Set')

plt.show()
```



Train Set

param_min_samples_split	1	5	10	50	100	500	1000
5	0.5683	0.6914	0.8157	0.9905	0.9985	1	1
10	0.5683	0.6914	0.8122	0.9847	0.996	0.9988	0.9988
20	0.5683	0.6912	0.8055	0.9759	0.992	0.995	0.9952
45	0.5683	0.6908	0.7975	0.9603	0.9797	0.985	0.9851
75	0.5683	0.6908	0.7913	0.9455	0.9687	0.9748	0.9746
100	0.5683	0.6894	0.7871	0.9361	0.9589	0.9658	0.9655
135	0.5683	0.6894	0.7837	0.9254	0.9475	0.9558	0.9554
270	0.5683	0.6894	0.7771	0.8909	0.9098	0.9183	0.9163
500	0.5683	0.6886	0.7568	0.8429	0.8533	0.8574	0.8588

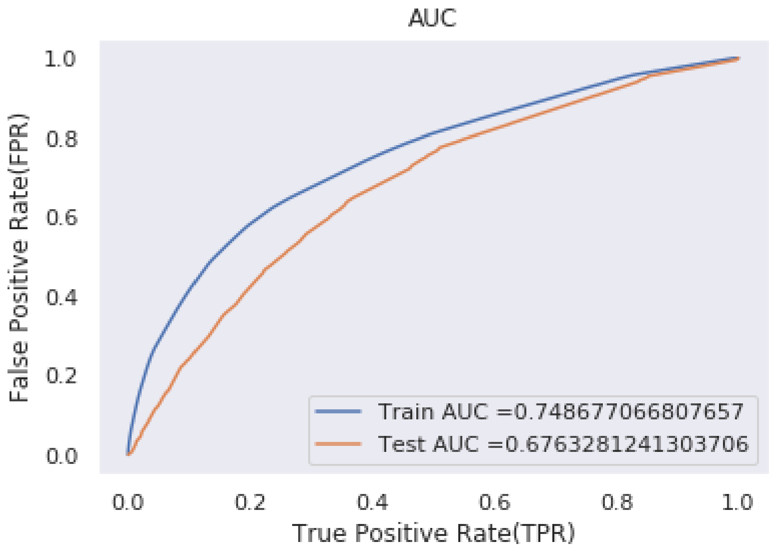
CV Set

param_min_samples_split	1	5	10	50	100	500	1000
5	0.5683	0.653	0.6578	0.5774	0.5626	0.5651	0.5603
10	0.5683	0.653	0.6562	0.5781	0.5647	0.5668	0.5652
20	0.5683	0.6527	0.6565	0.5832	0.5714	0.5678	0.5708
45	0.5683	0.6522	0.6566	0.5973	0.5818	0.5766	0.5755
75	0.5683	0.6521	0.6582	0.6027	0.5838	0.5801	0.5771
100	0.5683	0.6513	0.6613	0.6066	0.5869	0.5836	0.5828
135	0.5683	0.6513	0.6629	0.606	0.5892	0.5815	0.5791
270	0.5683	0.6513	0.6659	0.6182	0.6019	0.5876	0.5904
500	0.5683	0.6516	0.6707	0.6279	0.6211	0.6171	0.6122

Train model using the best hyper-parameter value

In [139]:

```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.m
from sklearn.metrics import roc_curve, auc
model = DecisionTreeClassifier(max_depth = 10, min_samples_split = 500,class_weight = 'bala
model.fit(X_tr2, y_train)
clfV2=DecisionTreeClassifier (class_weight = 'balanced',max_depth=3,min_samples_split=500)
# for visulation
clfV2.fit(X_tr2, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr2)
y_test_pred = batch_predict(model, X_te2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix -Train data

In [140]:

```
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24998866634136951 for threshold 0.329
[[ 3738  3688]
 [ 7959 33656]]
```

In [141]:

```
conf_matr_df_train_3 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

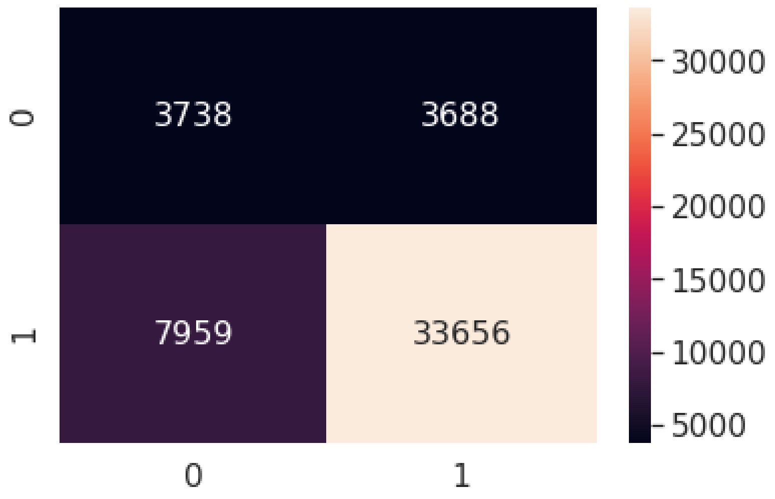
```
the maximum value of tpr*(1-fpr) 0.24998866634136951 for threshold 0.329
```

In [142]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_3, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[142]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e144124e0>
```



Test Data

In [143]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24995026120376243 for threshold 0.384
[[ 2691  2768]
 [ 7099 23494]]
```

In [144]:

```
conf_matr_df_test_4 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresho
```

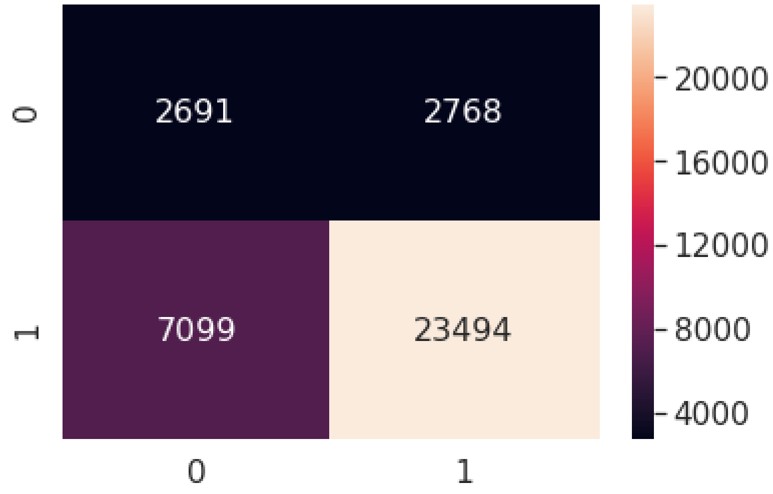
```
the maximum value of tpr*(1-fpr) 0.24995026120376243 for threshold 0.384
```

In [145]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_4, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[145]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e138a45f8>
```



Visualizing Decision Tree

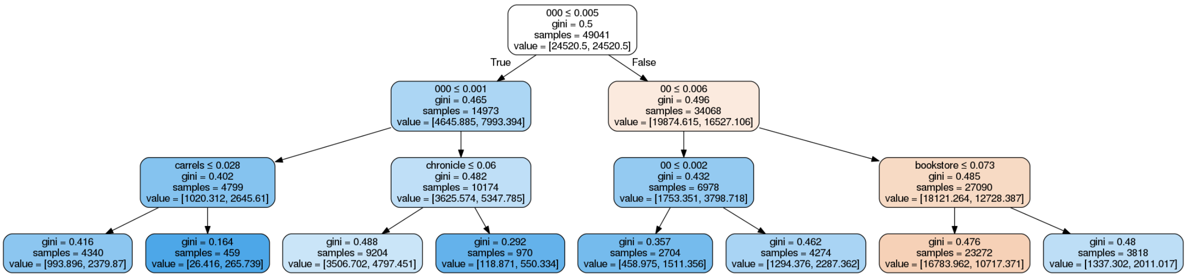
In [146]:

```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

In [147]:

```
dot_data = StringIO()

#dt_feat_names = list(X_test.columns)
#dt_target_names = [str(s) for s in [0,1]]
export_graphviz(clfV2, out_file=dot_data, filled=True, rounded=True, special_characters=Tru

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[147]:



False Positives Retrieval

In [148]:

```
fpi = []

for i in range(len(y_test)) :
    if (y_test.values[i] == 0) & (predictions1[i] == 1) :
        fpi.append(i)

fp_essay1 = []
for i in fpi :
    fp_essay1.append(X_test['clean_essays'].values[i])
```

In [149]:

```
len(fp_essay1)
```

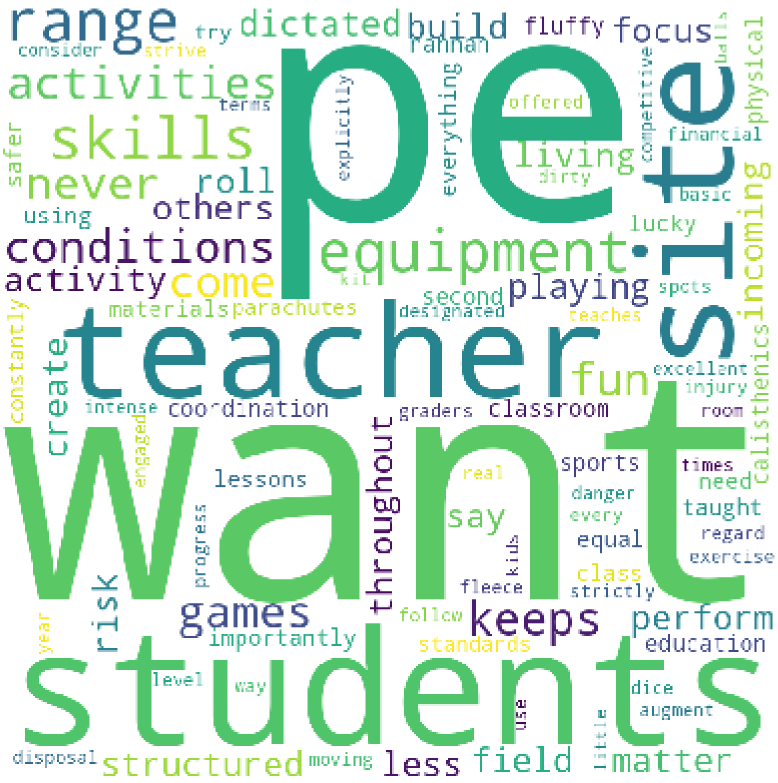
Out[149]:

```
2768
```

Word Cloud (False positives essay)

In [150]:

```
from wordcloud import WordCloud, STOPWORDS

comment_words = ' '
stopwords = set(STOPWORDS)

for val in fp_essay1 :
    val = str(val)
    tokens = val.split()
for i in range(len(tokens)):
    tokens[i] = tokens[i].lower()
for words in tokens :
    comment_words = comment_words + words + ' '

wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords = sto


plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



Building DataFrame of False Positives

In [151]:

```
cols = X_test.columns
X_test_falsePos1 = pd.DataFrame(columns=cols)
```

In [152]:

```
for i in fpi :
    X_test_falsePos1 = X_test_falsePos1.append(X_test.filter(items=[i], axis=0))
```

In [153]:

```
len(X_test_falsePos1)
```

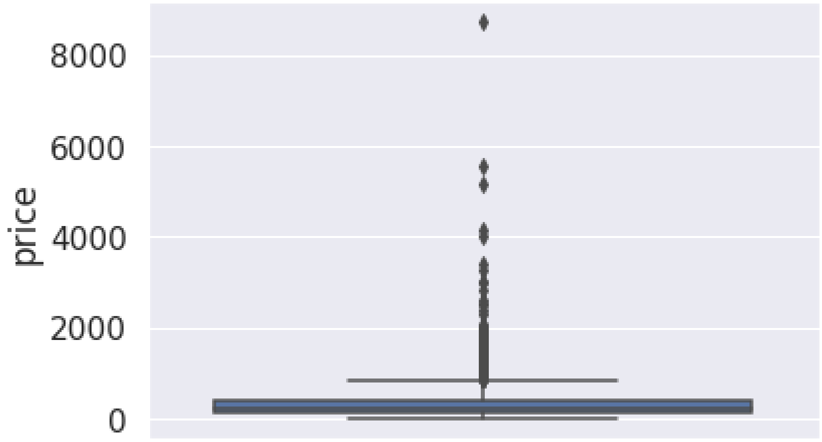
Out[153]:

```
2768
```

box plot for Price

In [154]:

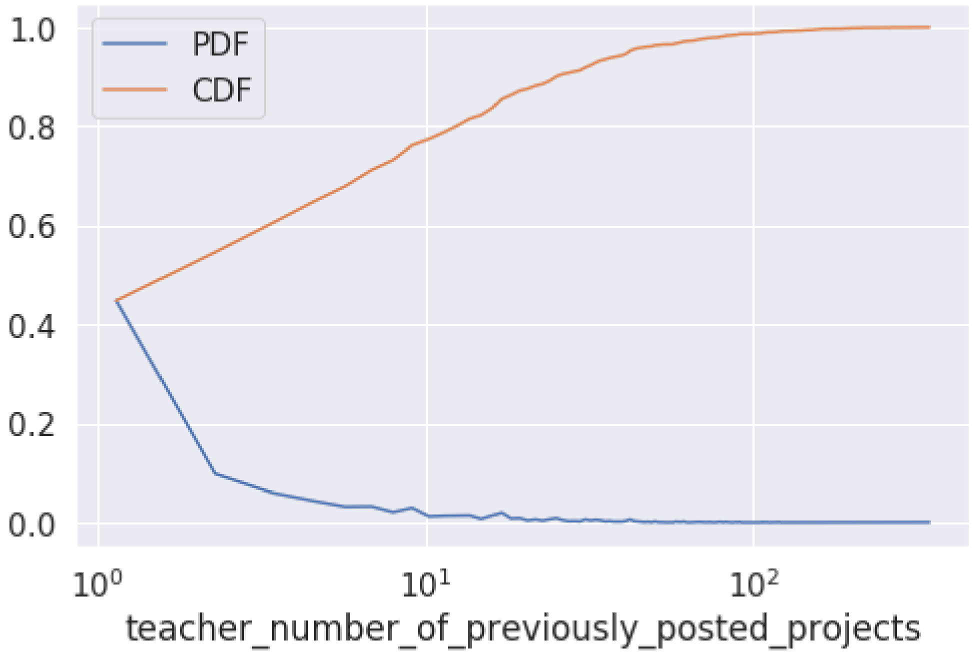
```
ax = sns.boxplot(y='price', data=X_test_falsePos1)
```



PDF on teacher_number_of_previously_posted_projects

In [155]:

```
plt.figure(figsize=(8,5))

counts, bin_edges = np.histogram(X_test_falsePos1['teacher_number_of_previously_posted_proj
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
pdfP, = plt.plot(bin_edges[1:], pdf)
cdfP, = plt.plot(bin_edges[1:], cdf)
plt.legend([pdfP, cdfP], ["PDF", "CDF"])
plt.xscale('log')
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



Set 3 : Categorical, Numerical features + Project_title(AVG W2V) + Preprocessed_essay (AVG W2V)

In [156]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,school_state_categori
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,school_state_categories
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,school_state_categories_one
```

In [157]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
```

```
Final Data matrix
(49041, 703) (49041,)
(24155, 703) (24155,)
(36052, 703) (36052,)
```

Gridsearch CV

In [158]:

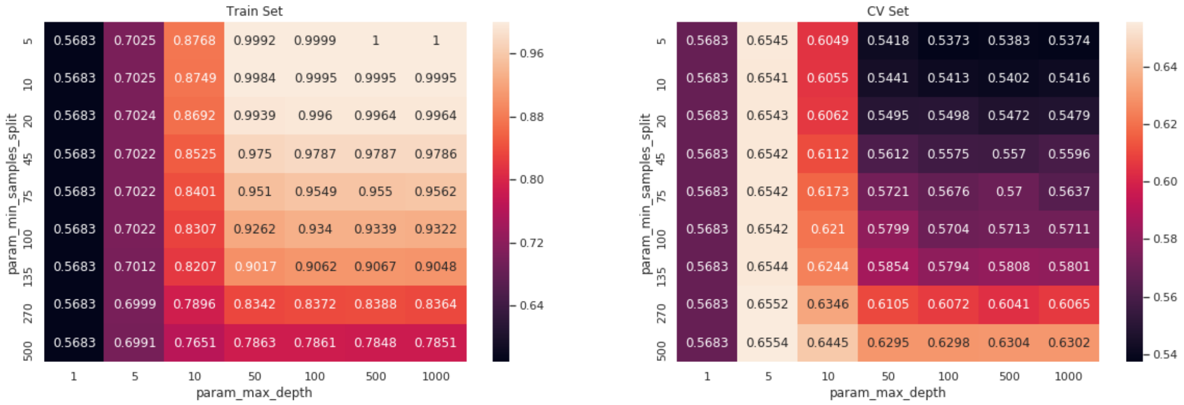
```
dt3 = DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10, 20,

clf3 = GridSearchCV(dt3, parameters, cv= 2, scoring='roc_auc',n_jobs=-1,return_train_score=
clf3.fit(X_tr, y_train)
```

Out[158]:

```
GridSearchCV(cv=2, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight='balanced',
                                              criterion='gini', max_depth=No
ne,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort=False, random_state=No
ne,
                                              splitter='best'),
             iid='warn', n_jobs=-1,
             param_grid={'max_depth': [1, 5, 10, 50, 100, 500, 1000],
                         'min_samples_split': [5, 10, 20, 45, 75, 100, 135,
270,
                                               500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [159]:

```
import seaborn as sns; sns.set()


max_scores3 = pd.DataFrame(clf3.cv_results_).groupby(['param_min_samples_split', 'param_max


fig, ax = plt.subplots(1,2, figsize=(20,6))

sns.heatmap(max_scores3.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores3.mean_test_score, annot = True, fmt='.4g', ax=ax[1])

ax[0].set_title('Train Set')
ax[1].set_title('CV Set')

plt.show()
```



Train Set

	1	5	10	50	100	500	1000
5	0.5683	0.7025	0.8768	0.9992	0.9999	1	1
10	0.5683	0.7025	0.8749	0.9984	0.9995	0.9995	0.9995
20	0.5683	0.7024	0.8692	0.9939	0.996	0.9964	0.9964
45	0.5683	0.7022	0.8525	0.975	0.9787	0.9787	0.9786
75	0.5683	0.7022	0.8401	0.951	0.9549	0.955	0.9562
100	0.5683	0.7022	0.8307	0.9262	0.934	0.9339	0.9322
135	0.5683	0.7012	0.8207	0.9017	0.9062	0.9067	0.9048
270	0.5683	0.6999	0.7896	0.8342	0.8372	0.8388	0.8364
500	0.5683	0.6991	0.7651	0.7863	0.7861	0.7848	0.7851

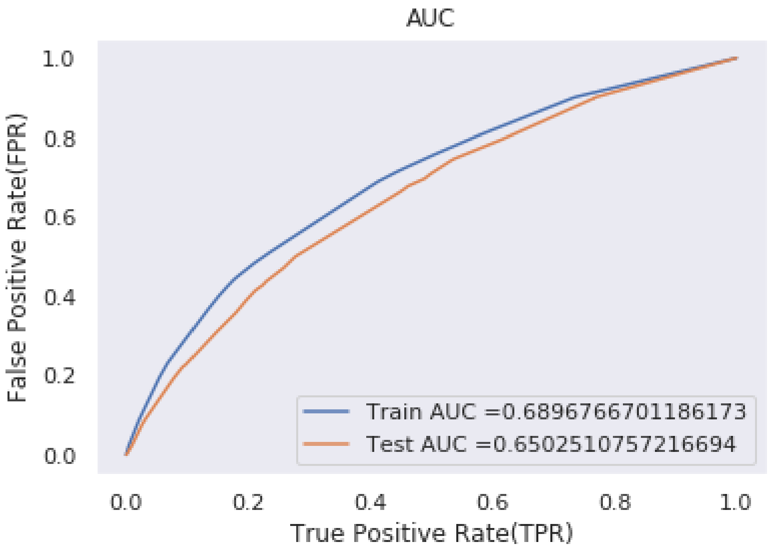
CV Set

	1	5	10	50	100	500	1000
5	0.5683	0.6545	0.6049	0.5418	0.5373	0.5383	0.5374
10	0.5683	0.6541	0.6055	0.5441	0.5413	0.5402	0.5416
20	0.5683	0.6543	0.6062	0.5495	0.5498	0.5472	0.5479
45	0.5683	0.6542	0.6112	0.5612	0.5575	0.557	0.5596
75	0.5683	0.6542	0.6173	0.5721	0.5676	0.57	0.5637
100	0.5683	0.6542	0.621	0.5799	0.5704	0.5713	0.5711
135	0.5683	0.6544	0.6244	0.5854	0.5794	0.5808	0.5801
270	0.5683	0.6552	0.6346	0.6105	0.6072	0.6041	0.6065
500	0.5683	0.6554	0.6445	0.6295	0.6298	0.6304	0.6302

B) Train the model using the best hyper parameter value

In [160]:

```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.m
from sklearn.metrics import roc_curve, auc
model = DecisionTreeClassifier(max_depth = 5, min_samples_split = 270,class_weight = 'balar
model.fit(X_tr, y_train)
clfV1=DecisionTreeClassifier (class_weight = 'balanced',max_depth=2,min_samples_split=500)
# for visulation
clfV1.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr)
y_test_pred = batch_predict(model, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



C) Confusion Matrix

Train data

In [161]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.2499999818661462 for threshold 0.411
[[ 3714  3712]
 [10258 31357]]
```

In [162]:

```
conf_matr_df_train_5 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

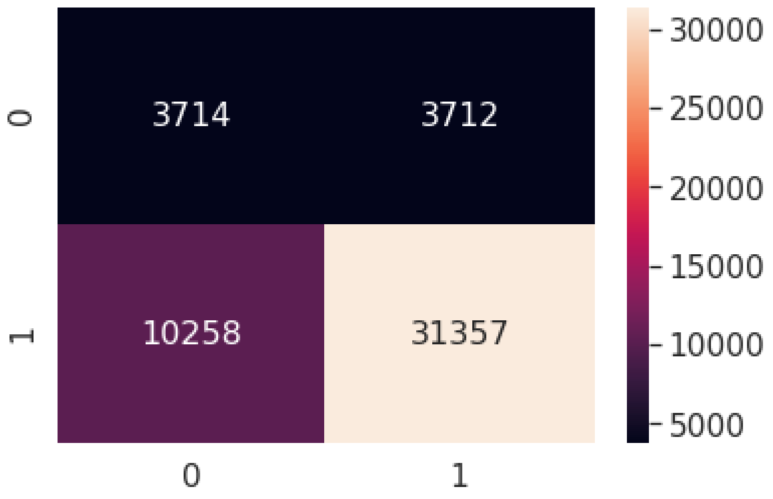
```
the maximum value of tpr*(1-fpr) 0.2499999818661462 for threshold 0.411
```

In [163]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_5, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[163]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e139735f8>
```



Test data

In [164]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 0.42
[[ 2729  2730]
 [ 8880 21713]]
```

In [165]:

```
conf_matr_df_test_6 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresho
```

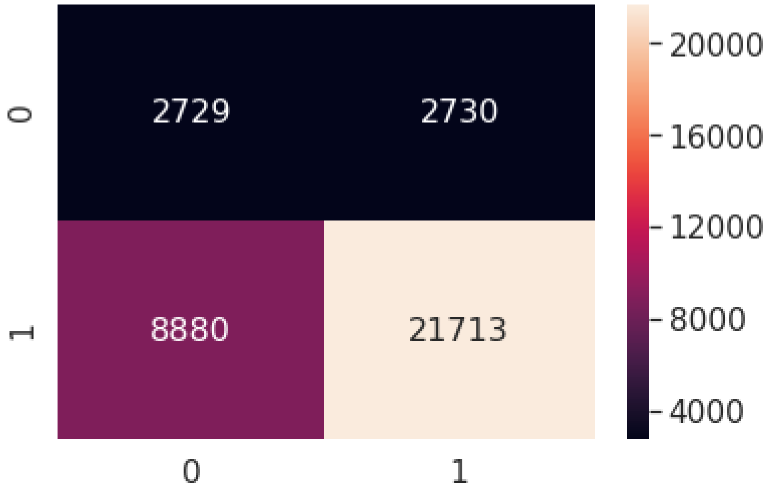
```
the maximum value of tpr*(1-fpr) 0.24999999161092998 for threshold 0.42
```

In [166]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_6, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[166]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e14349b38>
```



In [167]:

```
fpi = []

for i in range(len(y_test)) :
    if (y_test.values[i] == 0) & (predictions1[i] == 1) :
        fpi.append(i)

fp_essay1 = []
for i in fpi :
    fp_essay1.append(X_test['clean_essays'].values[i])
```

In [168]:

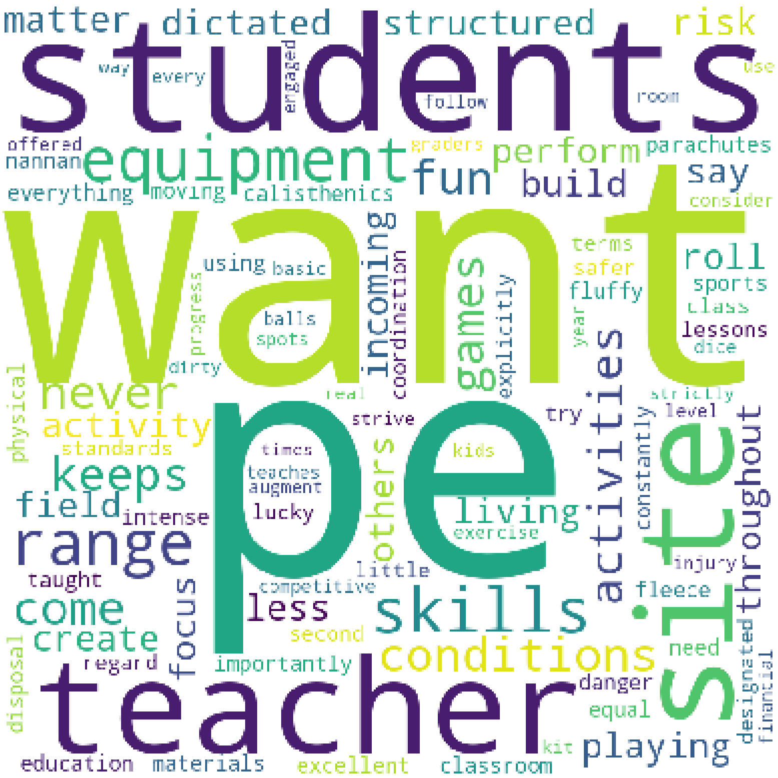
```
len(fp_essay1)
```

Out[168]:

```
2730
```

In [169]:

```
#Word cloud of essay
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay1 :
    val = str(val)
    tokens = val.split()
for i in range(len(tokens)):
    tokens[i] = tokens[i].lower()
for words in tokens :
    comment_words = comment_words + words + ' '
wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords =
stopwords,min_font_size = 10).generate(comment_words)
plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



In [170]:

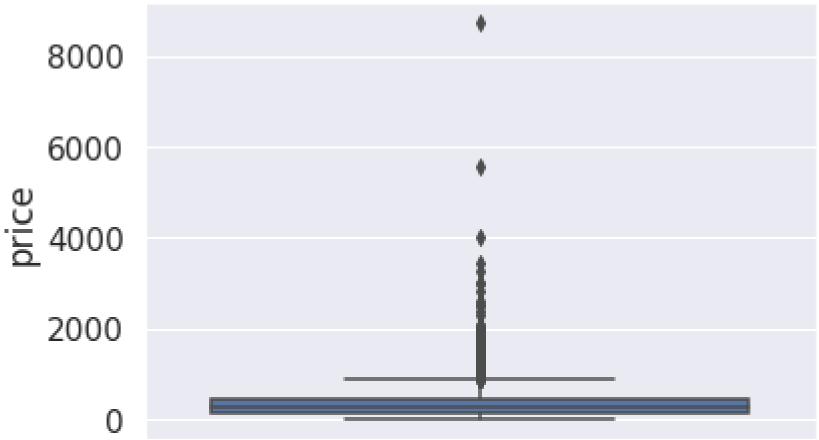
```
#DataFrame of False Positives
# first get the columns:
cols = X_test.columns
X_test_falsePos1 = pd.DataFrame(columns=cols)
# get the data of the false pisitives
for i in fpi : # (in fpi all the false positives data points indexes)
    X_test_falsePos1 = X_test_falsePos1.append(X_test.filter(items=[i], axis=0))
```

In [171]:

```
sns.boxplot(y='price', data=X_test_falsePos1)
```

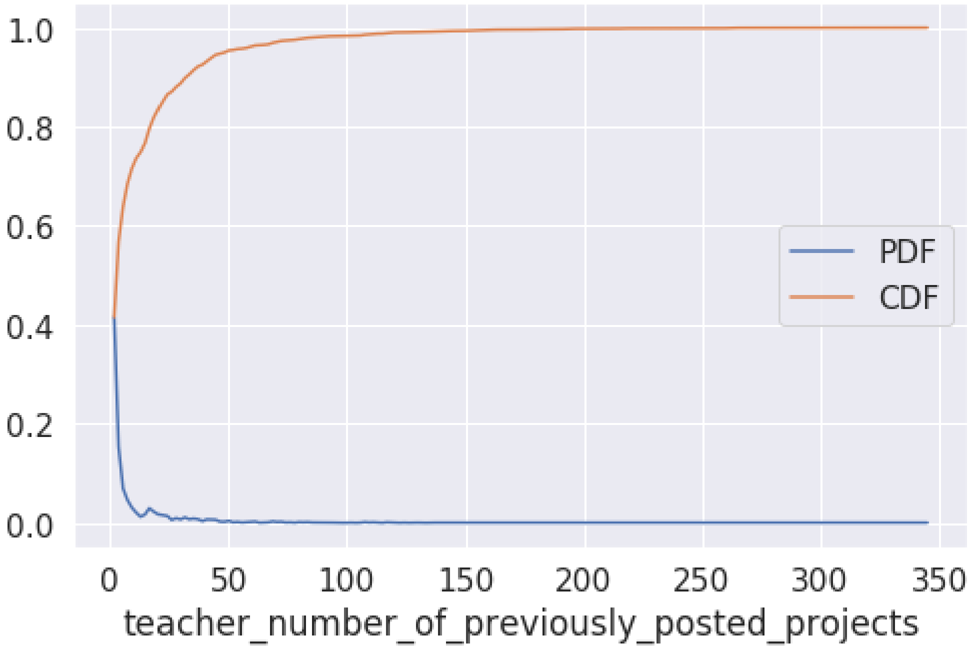
Out[171]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e13ff49e8>
```



In [172]:

```
#PDF (FP ,teacher_number_of_previously_posted_projects)
plt.figure(figsize=(8,5))
counts, bin_edges = np.histogram(X_test_falsePos1['teacher_number_of_previously_posted_proj
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
pdfP, = plt.plot(bin_edges[1:], pdf)
cdfP, = plt.plot(bin_edges[1:], cdf)
plt.legend([pdfP, cdfP], ["PDF", "CDF"])
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



In [ ]:

Set 4 : Categorical, Numerical features + Project_title(TFIDF W2V) + Preprocessed_essay (TFIDF W2V)

In [173]:

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

X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train,school_state_categori
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test,school_state_categories
X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv,school_state_categories_one
```

In [174]:

```
print("Final Data matrix")
print(X_tr.shape, y_train.shape)
print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
```

```
Final Data matrix
(49041, 703) (49041,)
(24155, 703) (24155,)
(36052, 703) (36052,)
```

GridSearchCV

In [175]:

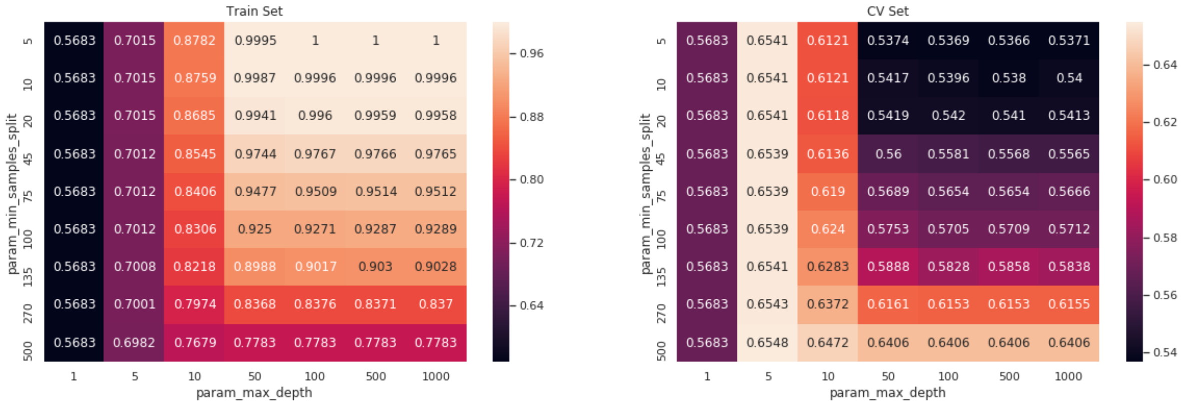
```
dt4 = DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10, 20,

clf4 = GridSearchCV(dt4, parameters, cv= 2, scoring='roc_auc',n_jobs=-1,return_train_score=
clf4.fit(X_tr, y_train)
```

Out[175]:

```
GridSearchCV(cv=2, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight='balanced',
                                              criterion='gini', max_depth=No
ne,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort=False, random_state=No
ne,
                                              splitter='best'),
             iid='warn', n_jobs=-1,
             param_grid={'max_depth': [1, 5, 10, 50, 100, 500, 1000],
                         'min_samples_split': [5, 10, 20, 45, 75, 100, 135,
270,
                                               500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [176]:

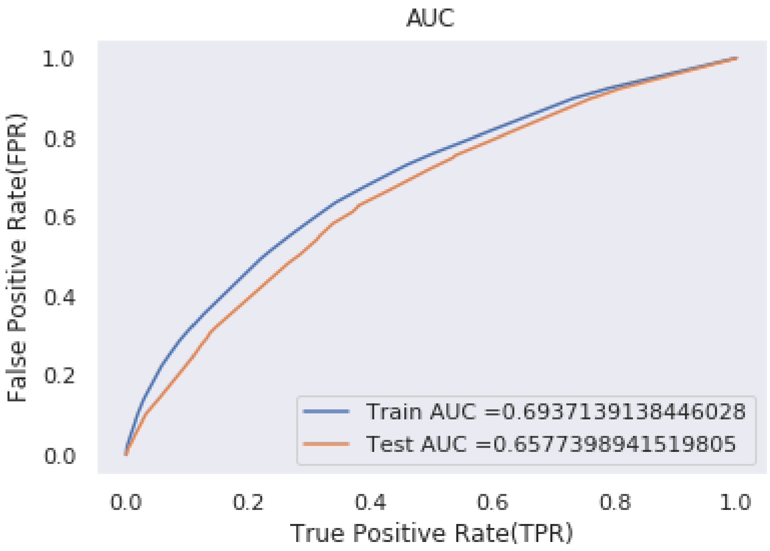
```
import seaborn as sns; sns.set()


max_scores4 = pd.DataFrame(clf4.cv_results_).groupby(['param_min_samples_split', 'param_max


fig, ax = plt.subplots(1,2, figsize=(20,6))

sns.heatmap(max_scores4.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores4.mean_test_score, annot = True, fmt='.4g', ax=ax[1])

ax[0].set_title('Train Set')
ax[1].set_title('CV Set')

plt.show()
```



Train the model using the best hyper parameter value

In [177]:

```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.m
from sklearn.metrics import roc_curve, auc
model = DecisionTreeClassifier(max_depth = 5, min_samples_split = 500,class_weight = 'balar
model.fit(X_tr, y_train)
clfV1=DecisionTreeClassifier (class_weight = 'balanced',max_depth=5,min_samples_split=500)
# for visulation
clfV1.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr)
y_test_pred = batch_predict(model, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix

In [178]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24999412463136592 for threshold 0.403
[[ 3731  3695]
 [10129 31486]]
```

In [179]:

```
conf_matr_df_train_7 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

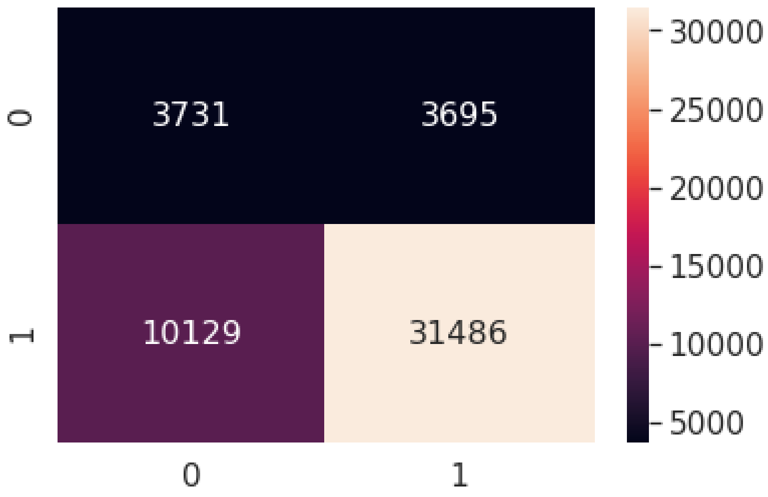
```
the maximum value of tpr*(1-fpr) 0.24999412463136592 for threshold 0.403
```

In [180]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_7, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[180]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e13e1dcf8>
```



In [181]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.24999858224716406 for threshold 0.444
[[ 2736  2723]
 [ 8510 22083]]
```

In [182]:

```
conf_matr_df_test_8 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresho
```

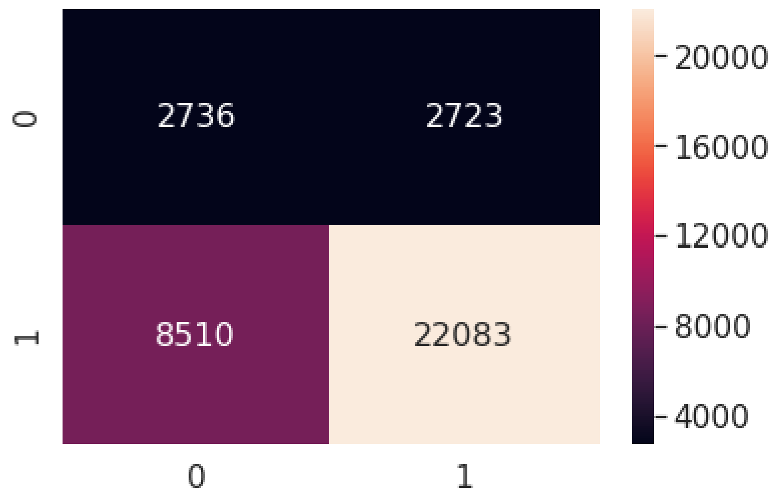
```
the maximum value of tpr*(1-fpr) 0.24999858224716406 for threshold 0.444
```

In [183]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_8, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[183]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e14310048>
```



In [184]:

```
#Analysis on the False positives
fpi = []
for i in range(len(y_test)) :
    if (y_test.values[i] == 0) & (predictions1[i] == 1) :
        fpi.append(i)
fp_essay1 = []
for i in fpi :
    fp_essay1.append(X_test['clean_essays'].values[i])
```

In [185]:

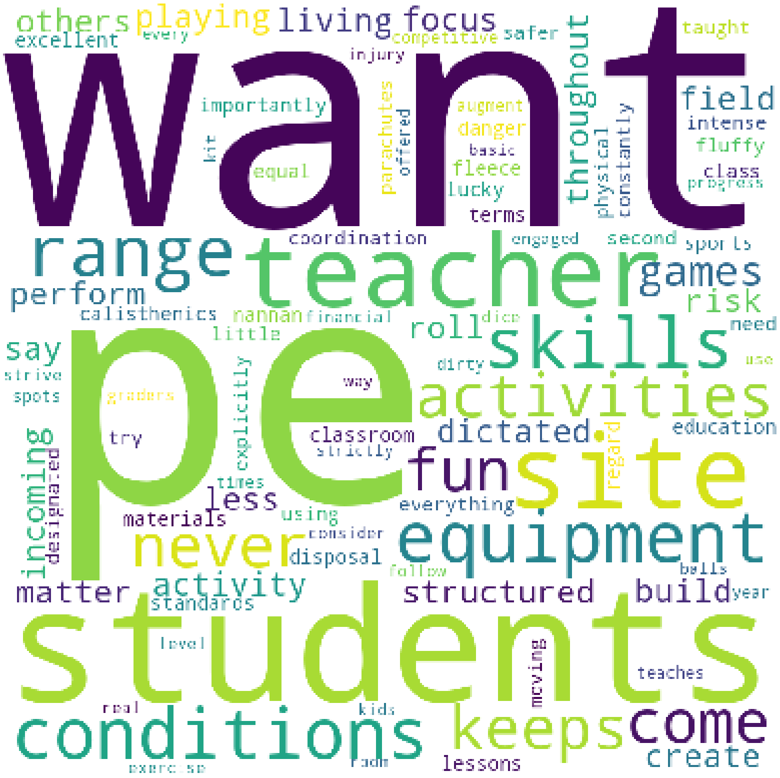
```
len(fp_essay1)
```

Out[185]:

```
2723
```

In [186]:

```
#WORD CLOUD OF ESSAY
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay1 :
    val = str(val)
    tokens = val.split()
for i in range(len(tokens)):
    tokens[i] = tokens[i].lower()
for words in tokens :
    comment_words = comment_words + words + ' '

wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords =stop
plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

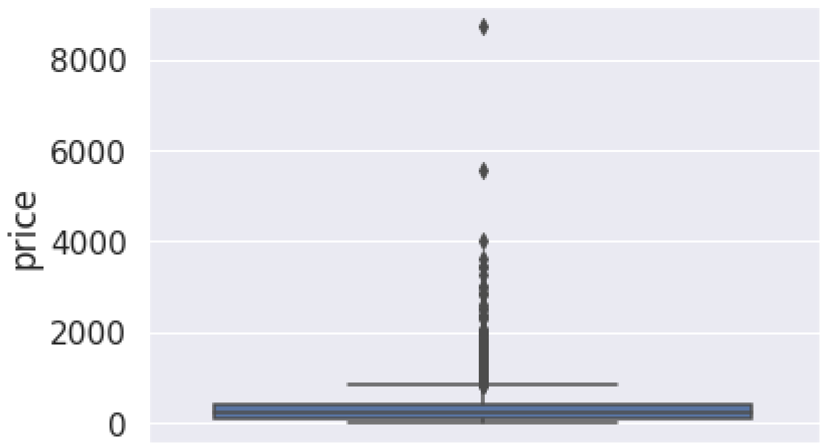


In [187]:

```
#Box Plot (FP 'price')
# first get the columns:
cols = X_test.columns
X_test_falsePos1 = pd.DataFrame(columns=cols)
# get the data of the false pisitives
for i in fpi : # (in fpi all the false positives data points indexes)
    X_test_falsePos1 = X_test_falsePos1.append(X_test.filter(items=[i], axis=0))
sns.boxplot(y='price', data=X_test_falsePos1)
```

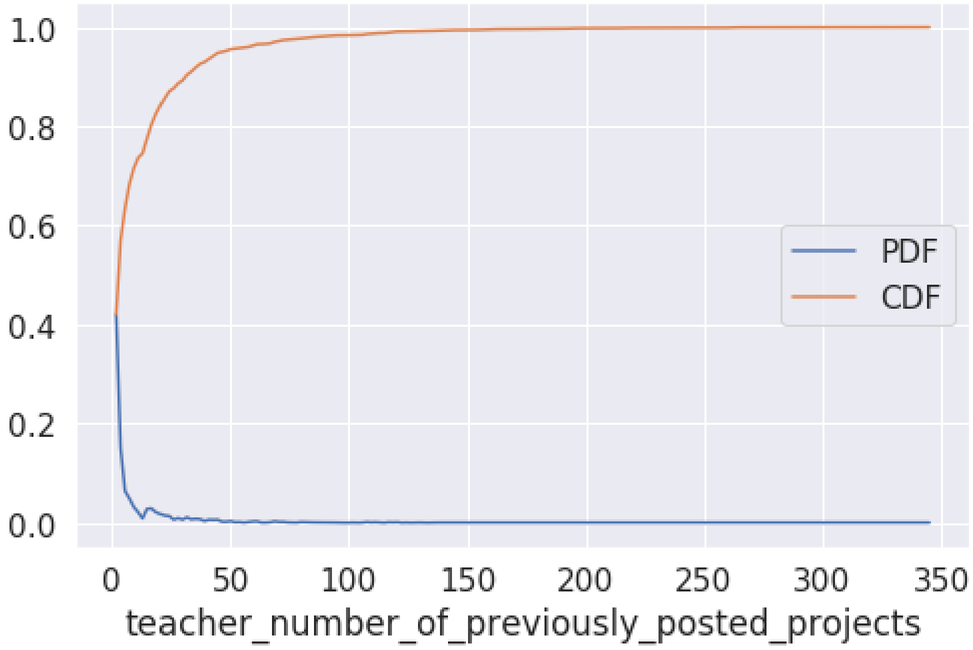
Out[187]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e13c5cf98>
```



In [188]:

```
#PDF (FP ,teacher_number_of_previously_posted_projects)
plt.figure(figsize=(8,5))
counts, bin_edges = np.histogram(X_test_falsePos1['teacher_number_of_previously_posted_proj
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
pdfP, = plt.plot(bin_edges[1:], pdf)
cdfP, = plt.plot(bin_edges[1:], cdf)
plt.legend([pdfP, cdfP], ["PDF", "CDF"])
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



In [ ]:

```

```

Set 5 Select 5k best features from features of Set 2 using feature_importances

In [189]:

```
#https://stackoverflow.com/questions/47111434/randomforestregressor-and-feature-importances
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV
def selectKImportance(model, X, k=5):
    return X[:,model.best_estimator_.feature_importances_.argsort()[::-1][:k]]
```

In [190]:

```
X_train5 = selectKImportance(clf2, X_tr2,5000)
X_test5 = selectKImportance(clf2, X_te2, 5000)
print(X_train5.shape)
print(X_test5.shape)
```

```
(49041, 5000)
(36052, 5000)
```

In [191]:

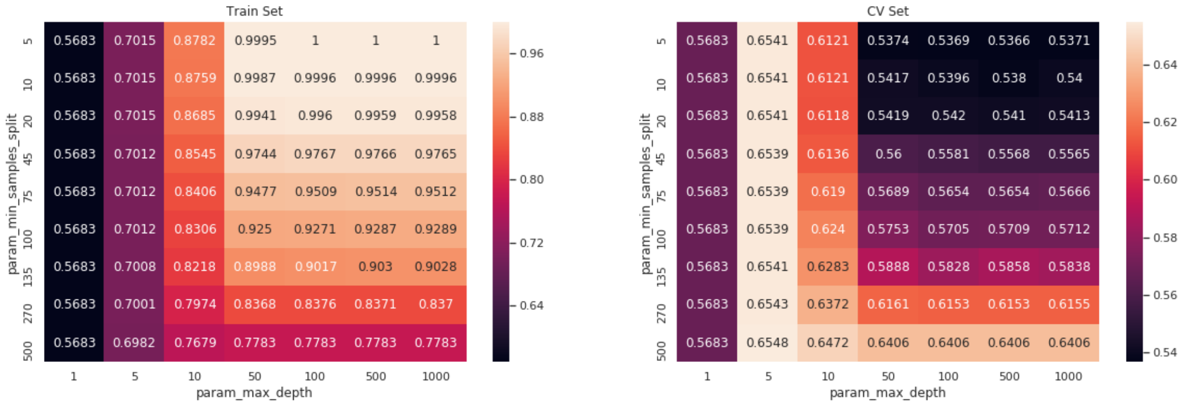
```
dt5 = DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10, 20,

clf5 = GridSearchCV(dt5, parameters, cv= 2, scoring='roc_auc',n_jobs=-1,return_train_score=
clf5.fit(X_tr2, y_train)
```

Out[191]:

```
GridSearchCV(cv=2, error_score='raise-deprecating',
             estimator=DecisionTreeClassifier(class_weight='balanced',
                                              criterion='gini', max_depth=No
ne,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort=False, random_state=No
ne,
                                              splitter='best'),
             iid='warn', n_jobs=-1,
             param_grid={'max_depth': [1, 5, 10, 50, 100, 500, 1000],
                         'min_samples_split': [5, 10, 20, 45, 75, 100, 135,
270,
                                               500]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='roc_auc', verbose=0)
```

In [192]:

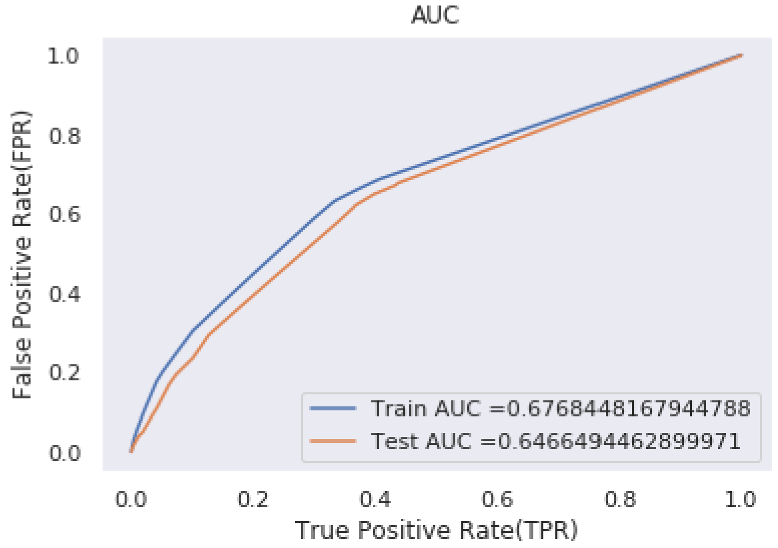
```
import seaborn as sns; sns.set()


max_scores5 = pd.DataFrame(clf4.cv_results_).groupby(['param_min_samples_split', 'param_max


fig, ax = plt.subplots(1,2, figsize=(20,6))

sns.heatmap(max_scores5.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores5.mean_test_score, annot = True, fmt='.4g', ax=ax[1])

ax[0].set_title('Train Set')
ax[1].set_title('CV Set')

plt.show()
```



Train Model using best Hyperparameter Value

In [193]:

```
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.m
from sklearn.metrics import roc_curve, auc
model = DecisionTreeClassifier(max_depth = 5, min_samples_split = 500,class_weight = 'balar
model.fit(X_tr2, y_train)
clfV1=DecisionTreeClassifier (class_weight = 'balanced',max_depth=5,min_samples_split=500)
# for visulation
clfV1.fit(X_tr2, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the p
# not the predicted outputs
y_train_pred = batch_predict(model, X_tr2)
y_test_pred = batch_predict(model, X_te2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
plt.plot(train_fpr, train_tpr, label="Train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="Test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.grid()
plt.show()
```



Confusion Matrix

In [194]:

```
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_fpr))
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.24210089328089218 for threshold 0.365
[[ 4373  3053]
 [12930 28685]]
```

In [195]:

```
conf_matr_df_train_9 = pd.DataFrame(confusion_matrix(y_train, predict(y_train_pred, tr_thre
```

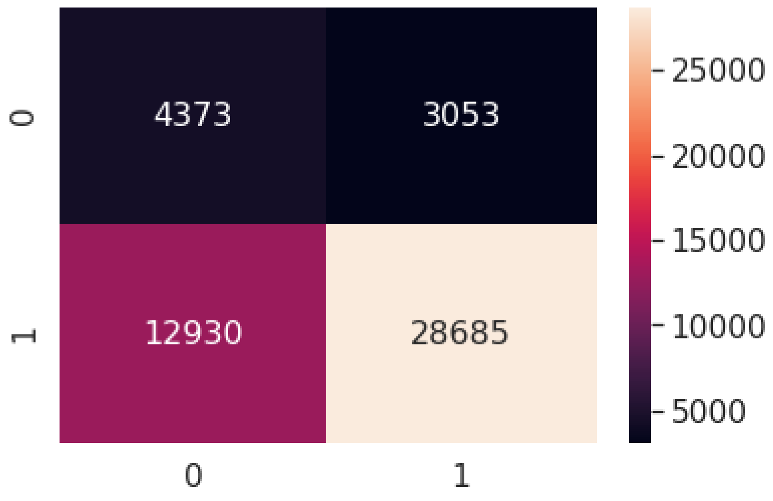
```
the maximum value of tpr*(1-fpr) 0.24210089328089218 for threshold 0.365
```

In [196]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train_9, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[196]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e13c579e8>
```



In [197]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))
```

```
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.2460636386128223 for threshold 0.365
[[ 3072  2387]
 [ 9879 20714]]
```

In [198]:

```
conf_matr_df_test_10 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred, tr_thresh
```

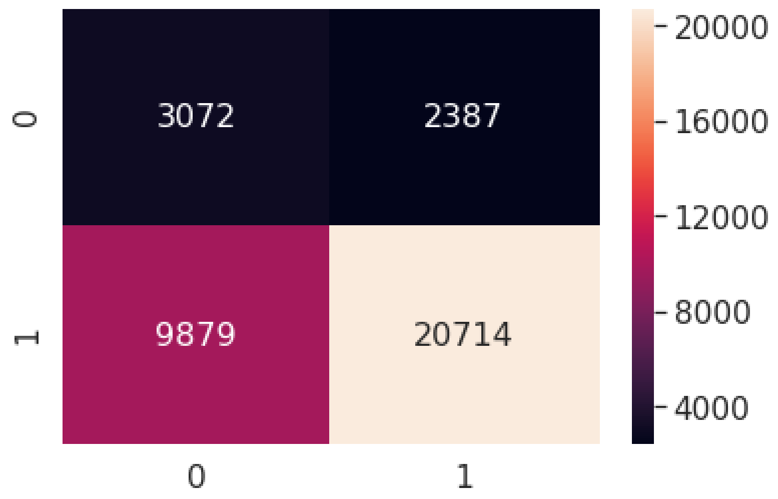
```
the maximum value of tpr*(1-fpr) 0.2460636386128223 for threshold 0.365
```

In [199]:

```
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test_10, annot=True,annot_kws={"size": 16}, fmt='g')
```

Out[199]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e143ea0f0>
```

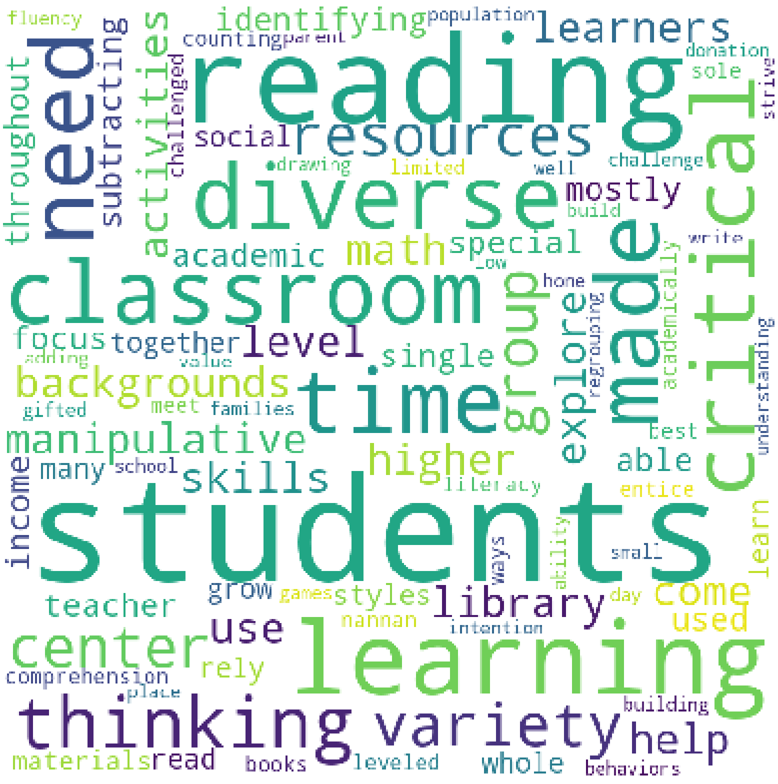


In [200]:

```
#Analysis on the False positives
fpi = []
for i in range(len(y_test)) :
    if (y_test.values[i] == 0) & (predictions1[i] == 1) :
        fpi.append(i)
fp_essay1 = []
for i in fpi :
    fp_essay1.append(X_test['clean_essays'].values[i])
```

In [201]:

```
#WORD CLOUD OF ESSAY
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay1 :
    val = str(val)
    tokens = val.split()
for i in range(len(tokens)):
    tokens[i] = tokens[i].lower()
for words in tokens :
    comment_words = comment_words + words + ' '

wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords =stop
plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

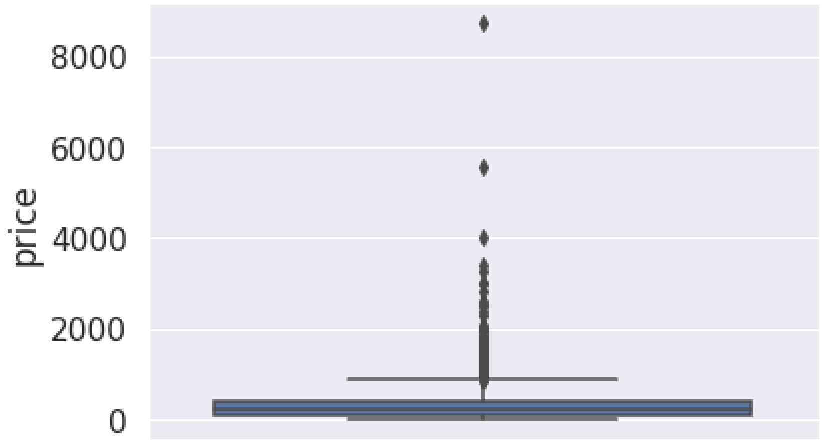


In [202]:

```
#Box Plot (FP 'price')
# first get the columns:
cols = X_test.columns
X_test_falsePos1 = pd.DataFrame(columns=cols)
# get the data of the false pisitives
for i in fpi : # (in fpi all the false positives data points indexes)
    X_test_falsePos1 = X_test_falsePos1.append(X_test.filter(items=[i], axis=0))
sns.boxplot(y='price', data=X_test_falsePos1)
```

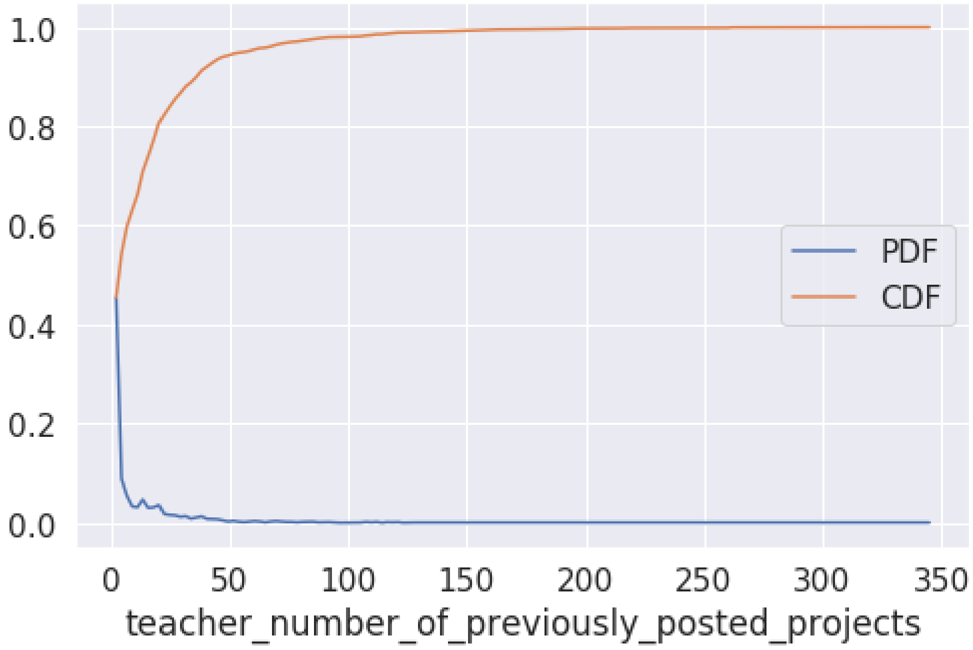
Out[202]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5e138b4630>
```



In [204]:

```
#PDF (FP ,teacher_number_of_previously_posted_projects)
plt.figure(figsize=(8,5))
counts, bin_edges = np.histogram(X_test_falsePos1['teacher_number_of_previously_posted_proj
pdf = counts/sum(counts)
cdf = np.cumsum(pdf)
pdfP, = plt.plot(bin_edges[1:], pdf)
cdfP, = plt.plot(bin_edges[1:], cdf)
plt.legend([pdfP, cdfP], ["PDF", "CDF"])
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



In [ ]:

In [205]:

```
from prettytable import PrettyTable

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

x = PrettyTable()
x.field_names = ["Vectorizer", "Max depth", "min samples Split", "AUC"]

x.add_row(["BOW", 100, 500, 0.674])
x.add_row(["TFIDF", 10, 500, 0.676])
x.add_row(["AVG W2V", 10, 270, 0.650])
x.add_row(["TFIDF W2V", 5, 500, 0.657])
x.add_row(["5k best features", 5, 500, 0.646])


print(x)
```

```
+------------------+-----------+-------------------+-------+
|    Vectorizer    | Max depth | min samples Split |  AUC  |
+------------------+-----------+-------------------+-------+
|       BOW        |    100    |        500        | 0.674 |
|      TFIDF       |     10    |        500        | 0.676 |
|     AVG W2V      |     10    |        270        |  0.65 |
|    TFIDF W2V     |     5     |        500        | 0.657 |
| 5k best features |     5     |        500        | 0.646 |
+------------------+-----------+-------------------+-------+
```

In [ ]: