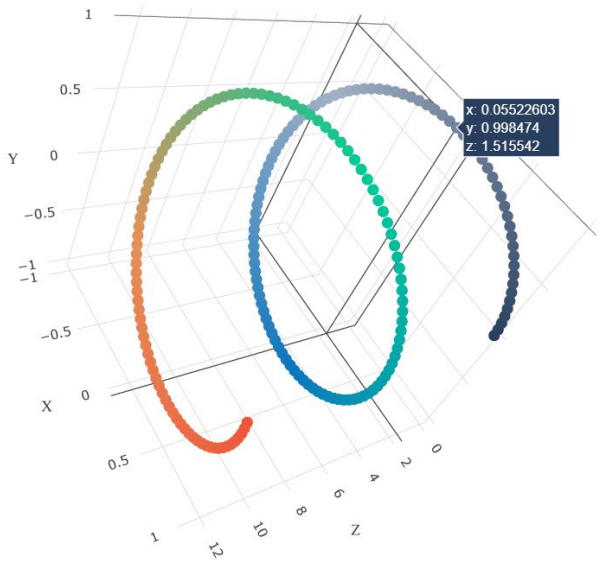
# Assignment 8: DT

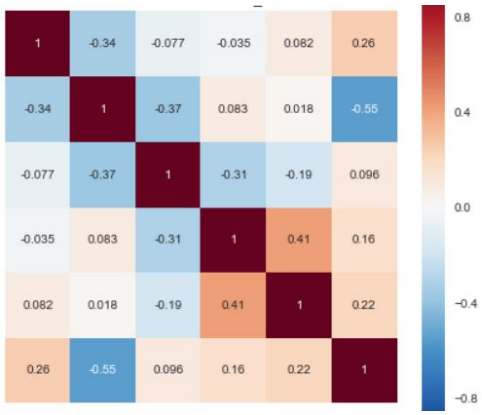
1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**
    - Set 1: categorical, numerical features + preprocessed_eassay (TFIDF)
    - Set 2: categorical, numerical features + preprocessed_eassay (TFIDF W2V)
2. **The hyper paramter tuning (best `depth` in range [1, 5, 10,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




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

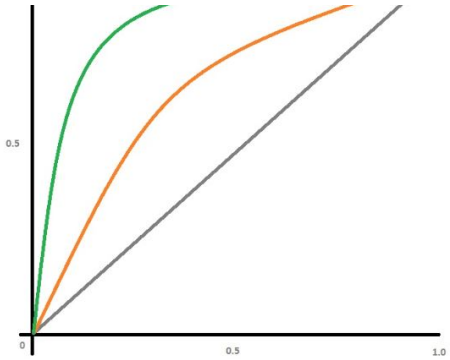
**or**

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




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

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




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

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

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

4. **Task 2:** For this task consider set-1 features. Select all the features which are having non-zero feature importance.You can get the feature importance using 'feature_importances_` (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html), discard the all other remaining features and then apply any of the model of you choice i.e. (Dession tree, Logistic Regression, Linear SVM), you need to do hyperparameter tuning corresponding to the model you selected and procedure in step 2 and step 3
   Note: when you want to find the feature importance make sure you don't use max_depth parameter keep it None.

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

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

In [1]:

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

import pandas as pd
import numpy as np
import nltk
import matplotlib.pyplot as plt
import seaborn as sns
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

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/

import pickle
from tqdm import tqdm
import os

from chart_studio import plotly
import plotly.offline as offline
```

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

# 1. Decision Tree

## 1.1 Loading Data

In [2]:

```
import pandas as pd


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

In [3]:

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

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

In [4]:

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

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

Out[4]:

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

## 1.2 Preprocessing of Data

# 1.2.1 Catgegorical Features

# Project Grade Category

In [5]:

```
project_data['project_grade_category'].value_counts()
```

Out[5]:

Out[5]:

```
Grades PreK-2    44225
Grades 3-5       37137
Grades 6-8       16923
Grades 9-12      10963
Name: project_grade_category, dtype: int64
```

In [6]:

```
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_'
)
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_'
)
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

Out[6]:

```
grades_prek_2    44225
grades_3_5       37137
grades_6_8       16923
grades_9_12      10963
Name: project_grade_category, dtype: int64
```

# project_subject_categories

In [7]:

```
project_data['project_subject_categories'].value_counts()
```

Out[7]:

```
Literacy & Language                           23655
Math & Science                                17072
Literacy & Language, Math & Science           14636
Health & Sports                               10177
Music & The Arts                               5180
Special Needs                                  4226
Literacy & Language, Special Needs             3961
Applied Learning                               3771
Math & Science, Literacy & Language            2289
Applied Learning, Literacy & Language          2191
History & Civics                               1851
Math & Science, Special Needs                  1840
Literacy & Language, Music & The Arts          1757
Math & Science, Music & The Arts               1642
Applied Learning, Special Needs                1467
History & Civics, Literacy & Language          1421
Health & Sports, Special Needs                 1391
Warmth, Care & Hunger                          1309
Math & Science, Applied Learning               1220
Applied Learning, Math & Science               1052
Literacy & Language, History & Civics           809
Health & Sports, Literacy & Language            803
Applied Learning, Music & The Arts              758
Math & Science, History & Civics                652
Literacy & Language, Applied Learning           636
Applied Learning, Health & Sports               608
Math & Science, Health & Sports                 414
History & Civics, Math & Science                322
History & Civics, Music & The Arts              312
Special Needs, Music & The Arts                 302
Health & Sports, Math & Science                 271
History & Civics, Special Needs                 252
Health & Sports, Applied Learning               192
Applied Learning, History & Civics              178
Health & Sports, Music & The Arts               155
Music & The Arts, Special Needs                 138
Literacy & Language, Health & Sports             72
Health & Sports, History & Civics                43
Special Needs, Health & Sports                   42
History & Civics, Applied Learning               42
Special Needs, Warmth, Care & Hunger             23
```

```
Special Needs, Warmth, Care & Hunger          25
Health & Sports, Warmth, Care & Hunger        23
Music & The Arts, Health & Sports             19
Music & The Arts, History & Civics            18
History & Civics, Health & Sports             13
Math & Science, Warmth, Care & Hunger         11
Music & The Arts, Applied Learning            10
Applied Learning, Warmth, Care & Hunger       10
Literacy & Language, Warmth, Care & Hunger     9
Music & The Arts, Warmth, Care & Hunger        2
History & Civics, Warmth, Care & Hunger        1
Name: project_subject_categories, dtype: int64
```

remove spaces, 'the' replace '&' with *'', and ',' with ''*

In [8]:

```
project_data['project_subject_categories'] =
project_data['project_subject_categories'].str.replace(' The ','')
project_data['project_subject_categories'] =
project_data['project_subject_categories'].str.replace(' ','')
project_data['project_subject_categories'] =
project_data['project_subject_categories'].str.replace('&','_')
project_data['project_subject_categories'] =
project_data['project_subject_categories'].str.replace(',','_')
project_data['project_subject_categories'] = project_data['project_subject_categories'].str.lower(
)
project_data['project_subject_categories'].value_counts()
```

Out[8]:

```
literacy_language                       23655
math_science                            17072
literacy_language_math_science          14636
health_sports                           10177
music_arts                               5180
specialneeds                             4226
literacy_language_specialneeds           3961
appliedlearning                          3771
math_science_literacy_language           2289
appliedlearning_literacy_language        2191
history_civics                           1851
math_science_specialneeds                1840
literacy_language_music_arts             1757
math_science_music_arts                  1642
appliedlearning_specialneeds             1467
history_civics_literacy_language         1421
health_sports_specialneeds               1391
warmth_care_hunger                       1309
math_science_appliedlearning             1220
appliedlearning_math_science             1052
literacy_language_history_civics          809
health_sports_literacy_language           803
appliedlearning_music_arts                758
math_science_history_civics               652
literacy_language_appliedlearning         636
appliedlearning_health_sports             608
math_science_health_sports                414
history_civics_math_science               322
history_civics_music_arts                 312
specialneeds_music_arts                   302
health_sports_math_science                271
history_civics_specialneeds               252
health_sports_appliedlearning             192
appliedlearning_history_civics            178
health_sports_music_arts                  155
music_arts_specialneeds                   138
literacy_language_health_sports            72
health_sports_history_civics               43
specialneeds_health_sports                 42
history_civics_appliedlearning             42
specialneeds_warmth_care_hunger            23
health_sports_warmth_care_hunger           23
music_arts_health_sports                   19
music_arts_history_civics                  18
history_civics_health_sports               13
```

```
history_civics_health_sports                15
math_science_warmth_care_hunger             11
appliedlearning_warmth_care_hunger          10
music_arts_appliedlearning                  10
literacy_language_warmth_care_hunger         9
music_arts_warmth_care_hunger                2
history_civics_warmth_care_hunger            1
Name: project_subject_categories, dtype: int64
```

# teacher_prefix

In [9]:

```
project_data['teacher_prefix'].value_counts()
```

Out[9]:

```
Mrs.       57269
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
Name: teacher_prefix, dtype: int64
```

In [10]:

```
print(project_data['teacher_prefix'].isnull().values.any())
print("number of nan values",project_data['teacher_prefix'].isnull().values.sum())
```

```
True
number of nan values 3
```

numebr of missing values are very less in number, we can replace it with Mrs. as most of the projects are submitted by Mrs.

In [11]:

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

In [12]:

```
project_data['teacher_prefix'].value_counts()
```

Out[12]:

```
Mrs.       57272
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
Name: teacher_prefix, dtype: int64
```

Remove '.' convert all the chars to small

In [13]:

```
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.replace('.','')
project_data['teacher_prefix'] = project_data['teacher_prefix'].str.lower()
project_data['teacher_prefix'].value_counts()
```

Out[13]:

```
mrs        57272
ms         38955
mr         10648
teacher     2360
dr            13
Name: teacher_prefix, dtype: int64
```

# project_subject_subcategories

In [14]:

```
project_data['project_subject_subcategories'].value_counts()
```

Out[14]:

```
Literacy                                     9486
Literacy, Mathematics                        8325
Literature & Writing, Mathematics            5923
Literacy, Literature & Writing               5571
Mathematics                                  5379
                                             ...
Parent Involvement, Warmth, Care & Hunger       1
Extracurricular, Financial Literacy             1
History & Geography, Warmth, Care & Hunger      1
Community Service, Gym & Fitness                1
Community Service, Financial Literacy           1
Name: project_subject_subcategories, Length: 401, dtype: int64
```

same process we did in project_subject_categories

In [15]:

```
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.
replace(' The ','')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.
replace(' ','')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.
replace('&','_')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.
replace(',','_')
project_data['project_subject_subcategories'] = project_data['project_subject_subcategories'].str.
lower()
project_data['project_subject_subcategories'].value_counts()
```

Out[15]:

```
literacy                                  9486
literacy_mathematics                      8325
literature_writing_mathematics            5923
literacy_literature_writing               5571
mathematics                               5379
                                          ...
extracurricular_financialliteracy            1
history_geography_warmth_care_hunger         1
college_careerprep_warmth_care_hunger        1
communityservice_financialliteracy           1
literature_writing_nutritioneducation        1
Name: project_subject_subcategories, Length: 401, dtype: int64
```

# school_state

In [16]:

```
project_data['school_state'].value_counts()
```

Out[16]:

```
CA    15388
TX     7396
NY     7318
FL     6185
NC     5091
IL     4350
GA     3963
SC     3936
MI     3161
```

```
PA     3109
IN     2620
MO     2576
OH     2467
LA     2394
MA     2389
WA     2334
OK     2276
NJ     2237
AZ     2147
VA     2045
WI     1827
AL     1762
UT     1731
TN     1688
CT     1663
MD     1514
NV     1367
MS     1323
KY     1304
OR     1242
MN     1208
CO     1111
AR     1049
ID      693
IA      666
KS      634
NM      557
DC      516
HI      507
ME      505
WV      503
NH      348
AK      345
DE      343
NE      309
SD      300
RI      285
MT      245
ND      143
WY       98
VT       80
Name: school_state, dtype: int64
```

convert all of them into small letters

In [17]:

```
project_data['school_state'] = project_data['school_state'].str.lower()
project_data['school_state'].value_counts()
```

Out[17]:

```
ca    15388
tx     7396
ny     7318
fl     6185
nc     5091
il     4350
ga     3963
sc     3936
mi     3161
pa     3109
in     2620
mo     2576
oh     2467
la     2394
ma     2389
wa     2334
ok     2276
nj     2237
az     2147
va     2045
wi     1827
al     1762
```

```
ut     1731
tn     1688
ct     1663
md     1514
nv     1367
ms     1323
ky     1304
or     1242
mn     1208
co     1111
ar     1049
id      693
ia      666
ks      634
nm      557
dc      516
hi      507
me      505
wv      503
nh      348
ak      345
de      343
ne      309
sd      300
ri      285
mt      245
nd      143
wy       98
vt       80
Name: school_state, dtype: int64
```

# project_title

In [18]:

```
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 [19]:

```
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
```

```
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [20]:

```
project_data['project_title'].head(5)
```

Out[20]:

```
0     Educational Support for English Learners at Home
1                Wanted: Projector for Hungry Learners
2    Soccer Equipment for AWESOME Middle School Stu...
3                               Techie Kindergarteners
4                               Interactive Math Tools
Name: project_title, dtype: object
```

In [21]:

```
print("printing some random reviews")
print(9, project_data['project_title'].values[9])
print(34, project_data['project_title'].values[34])
print(147, project_data['project_title'].values[147])
```

```
printing some random reviews
9 Just For the Love of Reading--\r\nPure Pleasure
34 \"Have A Ball!!!\"
147 Who needs a Chromebook?\r\nWE DO!!
```

In [22]:

```
from tqdm import tqdm
def preprocess_text(text_data):
    preprocessed_text = []
    # tqdm is for printing the status bar
    for sentance in tqdm(text_data):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('\\"', ' ')
        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_text.append(sent.lower().strip())
    return preprocessed_text
```

In [23]:

```
preprocessed_titles = preprocess_text(project_data['project_title'].values)
```

```
100%|████████████████████████████████████████████████████████████████████████████| 109248/109248
[00:04<00:00, 21887.53it/s]
```

In [24]:

```
print("printing some random reviews")
print(9, preprocessed_titles[9])
print(34, preprocessed_titles[34])
print(147, preprocessed_titles[147])
```

```
printing some random reviews
9 love reading pure pleasure
34 ball
```

147 needs chromebook

# Essay

In [25]:

```
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 [26]:

```
print("printing some random essay")
print(9, project_data['essay'].values[9])
print('-'*50)
print(34, project_data['essay'].values[34])
print('-'*50)
print(147, project_data['essay'].values[147])
```

```
printing some random essay
9 Over 95% of my students are on free or reduced lunch.  I have a few who are homeless, but despit
e that, they come to school with an eagerness to learn.  My students are inquisitive eager
learners who  embrace the challenge of not having great books and other resources  every day.
Many of them are not afforded the opportunity to engage with these big colorful pages of a book on
a regular basis at home and they don't travel to the public library.  \r\nIt is my duty as a
teacher to do all I can to provide each student an opportunity to succeed in every aspect of life.
\r\nReading is Fundamental! My students will read these books over and over again while boosting t
heir comprehension skills. These books will be used for read alouds, partner reading and for
Independent reading. \r\nThey will engage in reading to build their \"Love for Reading\" by
reading for pure enjoyment. They will be introduced to some new authors as well as some old
favorites. I want my students to be ready for the 21st Century and know the pleasure of holding a
good hard back book in hand. There's nothing like a good book to read!  \r\nMy students will soar
in Reading, and more because of your consideration and generous funding contribution. This will he
lp build stamina and prepare for 3rd grade. Thank you so much for reading our proposal!nannan
--------------------------------------------------
34 My students mainly come from extremely low-income families, and the majority of them come from
homes where both parents work full time. Most of my students are at school from 7:30 am to 6:00 pm
(2:30 to 6:00 pm in the after-school program), and they all receive free and reduced meals for bre
akfast and lunch. \r\n\r\n\r\nI want my students to feel  as comfortable in my classroom as they d
o at home. Many of my students take on multiple roles both at home as well as in school. They are
sometimes the caretakers of younger siblings, cooks, babysitters, academics, friends, and most of
all, they are developing who they are going to become as adults.  I consider it an essential part
of my job to model helping others gain knowledge in a positive manner. As a result, I have a commu
nity of students who love helping each other in and outside of the classroom. They consistently lo
ok for opportunities to support each other's learning in a kind and helpful way.I am excited to be
experimenting with alternative seating in my classroom this school year. Studies have shown that g
iving students the option of where they sit in a classroom increases focus as well as motivation.
\r\n\r\nBy allowing students choice in the classroom, they are able to explore and create in a wel
coming environment. Alternative classroom seating has been experimented with more frequently in re
cent years. I believe (along with many others), that every child learns differently. This does not
only apply to how multiplication is memorized, or a paper is written, but applies to the space in
which they are asked to work. I have had students in the past ask \"Can I work in the library? Can
I work on the carpet?\" My answer was always, \"As long as you're learning, you can work wherever
you want!\" \r\n\r\nWith the yoga balls and the lap-desks, I will be able to increase the options
for seating in my classroom and expand its imaginable space.nannan
--------------------------------------------------
147 My students are eager to learn and make their mark on the world.\r\n\r\nThey come from a Title
1 school and need extra love.\r\n\r\nMy fourth grade students are in a high poverty area and still
come to school every day to get their education. I am trying to make it fun and educational for th
em so they can get the most out of their schooling. I created a caring environment for the student
s to bloom! They deserve the best.\r\nThank you!\r\nI am requesting 1 Chromebook to access online
interventions, differentiate instruction, and get extra practice. The Chromebook will be used to s
upplement ELA and math instruction. Students will play ELA and math games that are engaging and fu
n, as well as participate in assignments online. This in turn will help my students improve their
skills. Having a Chromebook in the classroom would not only allow students to use the programs at
their own pace, but would ensure more students are getting adequate time to use the programs. The
online programs have been especially beneficial to my students with special needs. They are able t
o work at their level as well as be challenged with some different materials. This is making these
students more confident in their abilities.\r\n\r\nThe Chromebook would allow my students to have
daily access to computers and increase their computing skills.\r\nThis will change their lives for
the better as they become more successful in school. Having access to technology in the classroom
```

would help bridge the achievement gap.nannan

In [27]:

```
preprocessed_essays = preprocess_text(project_data['essay'].values)
```

```
100%|████████████████████████████████████████████████████████████| 109248/109248 [01:16<00:00, 1435.25it/s]
```

In [28]:

```
print("printing some random essay")
print(9, preprocessed_essays[9])
print('-'*50)
print(34, preprocessed_essays[34])
print('-'*50)
print(147, preprocessed_essays[147])
```

```
printing some random essay
9 95 students free reduced lunch homeless despite come school eagerness learn students inquisitive eager learners embrace challenge not great books resources every day many not afforded opportunity engage big colorful pages book regular basis home not travel public library duty teacher provide s tudent opportunity succeed every aspect life reading fundamental students read books boosting comp rehension skills books used read alouds partner reading independent reading engage reading build l ove reading reading pure enjoyment introduced new authors well old favorites want students ready 2 1st century know pleasure holding good hard back book hand nothing like good book read students so ar reading consideration generous funding contribution help build stamina prepare 3rd grade thank much reading proposal nannan
--------------------------------------------------
34 students mainly come extremely low income families majority come homes parents work full time s tudents school 7 30 6 00 pm 2 30 6 00 pm school program receive free reduced meals breakfast lunch want students feel comfortable classroom home many students take multiple roles home well school s ometimes caretakers younger siblings cooks babysitters academics friends developing going become a dults consider essential part job model helping others gain knowledge positive manner result commu nity students love helping outside classroom consistently look opportunities support learning kind helpful way excited experimenting alternative seating classroom school year studies shown giving s tudents option sit classroom increases focus well motivation allowing students choice classroom ab le explore create welcoming environment alternative classroom seating experimented frequently rece nt years believe along many others every child learns differently not apply multiplication memorized paper written applies space asked work students past ask work library work carpet answer always long learning work wherever want yoga balls lap desks able increase options seating classroom expand imaginable space nannan
--------------------------------------------------
147 students eager learn make mark world come title 1 school need extra love fourth grade students high poverty area still come school every day get education trying make fun educational get school ing created caring environment students bloom deserve best thank requesting 1 chromebook access on line interventions differentiate instruction get extra practice chromebook used supplement ela mat h instruction students play ela math games engaging fun well participate assignments online turn h elp students improve skills chromebook classroom would not allow students use programs pace would ensure students getting adequate time use programs online programs especially beneficial students special needs able work level well challenged different materials making students confident abilities chromebook would allow students daily access computers increase computing skills change lives better become successful school access technology classroom would help bridge achievement ga p nannan
```

# Price

In [29]:

```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[29]:

| | id | price | quantity |
|---|---|---|---|
| 0 | p000001 | 459.56 | 7 |
| 1 | p000002 | 515.89 | 21 |

In [30]:

```
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [31]:

```
project_data['price'].head()
```

Out[31]:

```
0    154.60
1    299.00
2    516.85
3    232.90
4     67.98
Name: price, dtype: float64
```

# Applying StandardScaler

In [32]:

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

In [33]:

```
project_data['std_price'].head()
```

Out[33]:

```
0   -0.390533
1    0.002396
2    0.595191
3   -0.177469
4   -0.626236
Name: std_price, dtype: float64
```

# Applying MinMaxScaler

In [34]:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(project_data['price'].values.reshape(-1, 1))
project_data['nrm_price']=scaler.transform(project_data['price'].values.reshape(-1, 1))
```

In [35]:

```
project_data['nrm_price'].head()
```

Out[35]:

```
0    0.015397
1    0.029839
2    0.051628
3    0.023228
4    0.006733
Name: nrm_price, dtype: float64
```

# Title_counts

In [36]:

```
title_number_words=[]

for x in project_data['project_title']:
    y=len(x.split())
    title_number_words.append(y)
```

In [37]:

```
project_data['title_number_words']=title_number_words
```

# Essay_counts

In [38]:

```
essay_number_words=[]

for x in project_data['essay']:
    y=len(x.split())
    essay_number_words.append(y)
```

In [39]:

```
project_data['essay_number_words']=essay_number_words
```

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

In [40]:

```
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_approved'], test_size=0.33, stratify = project_data['project_is_approved'
],random_state=0)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train,
random_state=0)
```

In [41]:

```
X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

### 1.4 Make Data Model Ready: encoding eassay, and project_title

# 1.4.1 encoding Text features: Essay-BOW

In [42]:

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

print("="*100)


vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['essay'].values) # fit has to happen only on train data
```

```
# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer.transform(X_train['essay'].values)
X_cv_essay_bow = vectorizer.transform(X_cv['essay'].values)
X_test_essay_bow = vectorizer.transform(X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)



print("NOTE: THE NUMBER OF COLUMNS IN EACH OF THE VECTOR WONT BE SAME")
```

```
(49041, 23) (49041,)
(24155, 23) (24155,)
(36052, 23) (36052,)
====================================================================================================

After vectorizations
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
====================================================================================================

NOTE: THE NUMBER OF COLUMNS IN EACH OF THE VECTOR WONT BE SAME
```

In [43]:

```
vectorizer_essay_bow = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer_essay_bow.fit(X_train['essay'].values)
```

Out[43]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=5000, min_df=10,
                ngram_range=(1, 4), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)
```

## 1.4.2 encoding Text features: Essay-TFIDF

In [44]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

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

print("="*100)


vectorizer_essay_Tfidf = TfidfVectorizer(min_df=10, max_features=5000)
vectorizer_essay_Tfidf.fit(X_train['essay'].values) # fit has to happen only on train data

# we use the fitted TfidfVectorizer to convert the text to vector
X_train_essay_Tfidf = vectorizer_essay_Tfidf.transform(X_train['essay'].values)
X_cv_essay_Tfidf = vectorizer_essay_Tfidf.transform(X_cv['essay'].values)
X_test_essay_Tfidf = vectorizer_essay_Tfidf.transform(X_test['essay'].values)

print("After vectorizations")
print(X_train_essay_Tfidf.shape, y_train.shape)
print(X_cv_essay_Tfidf.shape, y_cv.shape)
print(X_test_essay_Tfidf.shape, y_test.shape)
print("="*100)



print("NOTE: THE NUMBER OF COLUMNS IN EACH OF THE VECTOR WONT BE SAME")
```

```
print( NOTE: THE NUMBER OF COLUMNS IN EACH OF THE VECTOR WONT BE SAME )
```

```
(49041, 23) (49041,)
(24155, 23) (24155,)
(36052, 23) (36052,)
==========================================================================================================

After vectorizations
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
==========================================================================================================

NOTE: THE NUMBER OF COLUMNS IN EACH OF THE VECTOR WONT BE SAME
```

## 1.4.3 encoding Text features: Essay-Avg W2Vec

In [45]:

```
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [46]:

```
avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay'].values): # 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]))
print(avg_w2v_vectors_train[0])
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 49041/49041
[00:28<00:00, 1708.47it/s]

49041
300
[-4.81504624e-02 -6.96376980e-02 -6.75092919e-02 -1.67244788e-01
  4.55009062e-02 -3.63718087e-02 -3.53038855e+00  2.14568355e-01
  6.42519884e-02 -1.55032998e-01  1.46054072e-01  5.10826074e-02
 -7.30481503e-02 -1.26345213e-01 -7.93205231e-02 -1.15440792e-01
 -8.66065815e-02 -1.02057400e-01  1.06090202e-01  1.02253513e-01
  5.48120313e-02  1.69298145e-02 -5.92954382e-02  4.93544249e-02
 -3.66171408e-02 -2.55012139e-02  2.98302699e-02 -1.62567702e-01
 -1.12650346e-01 -1.13204887e-01 -2.36546064e-01 -8.67428994e-02
  4.54593665e-02 -3.21474497e-02 -1.36306199e-01 -7.75393353e-03
 -1.20910971e-02 -1.09602115e-01  7.57303966e-02 -8.40817108e-02
 -4.87067647e-02  8.93267879e-02  3.78309121e-02 -2.01257235e-01
 -5.31506647e-02 -3.52255613e-02  7.79028087e-02 -1.66601382e-01
 -1.37675665e-01  2.45329587e-02  3.90471532e-02  9.41483717e-02
 -2.34469965e-02 -4.79321272e-02  6.30726364e-02 -1.14433614e-01
  7.98227150e-02 -2.46039087e-02 -6.46417613e-02  7.81061341e-02
 -4.15729376e-02  6.70791341e-02  3.76362000e-02 -1.13544193e-01
 -6.72277283e-02  1.91462210e-01  7.99065376e-02  2.71086156e-02
  1.85660218e-01 -1.07133983e-01 -2.13331237e-01  4.78215087e-03
 -2.15113983e-02 -7.04178780e-02  1.97596358e-02 -2.40324114e-01
  1.15396225e-01  1.36073306e-02  1.35610002e-01 -1.08026359e-01
  5.86882954e-02 -5.66574209e-01 -9.28727668e-02 -1.58789284e-01
  4.39962448e-02  6.82431896e-02  3.90769355e-02 -1.01018818e-01
  1.09533990e-01 -2.19253543e-02  2.27211293e-02 -8.12973162e-02
 -1.69998046e-02  5.97638514e-02 -4.08562543e-03 -2.42898780e-01
 -2.63798549e+00 -6.15275682e-02  1.09555636e-01  8.76857382e-02
 -6.66988497e-02  1.52541442e-01  2.02361288e-01  7.03569561e-03
```

```
 -7.68750156e-02 -1.29391122e-02  1.11750587e-01 -1.49025734e-01
 -5.86507260e-02  5.36100376e-02 -1.90672379e-02  9.78639792e-02
  5.68016630e-02  1.80258873e-01 -1.13963705e-02  1.88951042e-02
 -2.80878220e-02 -3.18960116e-03  1.11103035e-01  5.75805751e-02
 -1.20890199e-01 -2.64726624e-02 -9.53153121e-04 -1.59678561e-01
  1.94861954e-02  1.42859191e-03 -2.73734854e-02 -7.46366746e-02
 -3.07473064e-03  7.89804058e-02  5.45301310e-02  2.38508087e-02
 -1.69581457e-02 -1.11752997e-01  8.23222439e-02  2.16516566e-02
  7.75765044e-02 -8.22167457e-03  2.44237079e-01  2.65827249e-01
  7.71381162e-02  3.27617769e-02 -1.29356396e-02 -1.92105087e-02
 -1.35399295e-02 -6.78832769e-02  8.06506538e-02 -3.90906432e-02
  3.30163775e-01  1.31624908e-01 -3.75681723e-02 -5.50561179e-02
  1.71931521e-02 -2.32684042e-02  3.28931538e-02 -7.48625055e-02
  4.74752607e-02  1.72015029e-02 -1.32693854e-01 -5.45671913e-02
  1.10370410e-01 -9.00208931e-02 -2.12085431e-02 -5.85889173e-02
 -7.28783214e-02  4.44754896e-02 -4.13681029e-02  9.21154468e-02
  1.35107873e-01 -1.00834398e-01 -3.16182254e-02  4.04800434e-02
 -7.59186156e-02 -8.17157844e-02 -1.05212967e-01  2.44601882e-01
 -1.18829621e-01 -9.51923069e-02 -9.24276301e-03 -1.11107040e-02
  1.23421866e-01  1.09983355e-01 -9.37664578e-02 -9.54882913e-02
  7.92411896e-02 -1.84819451e-01  3.50804532e-02  2.09569422e-02
  1.41346873e-01 -4.35895260e-02 -2.45048555e-03 -4.78310919e-02
  1.37811017e-02 -2.97323318e-03  4.56390058e-02 -1.13876612e-01
  6.78940231e-02  1.06860070e-01  1.33323577e-01  6.56578630e-03
  4.57764376e-02 -4.92224211e-02  3.13370545e-02 -1.32662428e-02
  1.50239828e-02  1.33395416e-01  3.90383873e-02 -1.80675723e-02
  1.29137161e-01 -2.17786908e-02  3.94724682e-02 -4.99195104e-02
 -5.70579156e-02 -7.40721543e-02 -7.68729844e-02 -3.99187087e-02
 -5.76343370e-02 -8.62897220e-02 -2.56204173e-02  1.50177152e-01
 -9.99371260e-02 -8.01535665e-02 -4.93496139e-02  3.05870514e-02
 -2.59842009e+00  6.66491549e-02 -5.22028150e-02 -2.01808295e-02
  2.51514480e-02 -8.22859650e-02  8.99183445e-02  3.96558555e-02
  2.21633694e-02 -7.50365873e-02 -1.18931557e-01  1.34325698e-01
 -5.61449942e-03 -8.52783624e-02 -6.74389364e-02  9.36177812e-02
 -1.44942909e-01  8.64188399e-02 -1.45114104e-01  3.99279266e-02
  2.01994763e-02 -6.53910809e-02 -5.26783376e-02 -5.35820266e-02
 -8.37511017e-02  2.11062491e-02 -6.39851971e-02 -8.15694046e-03
  1.28252954e-02 -1.38454318e-02  3.31428497e-02 -1.30689342e-02
  6.37582213e-02 -1.10993489e-01  1.16859814e-01  5.08639769e-03
  1.13039100e-01  3.48427746e-03 -4.32839555e-02  5.93400526e-02
  1.20318072e-01 -1.37683451e-01 -2.70569246e-01 -8.19515543e-02
  1.58294725e-01 -1.07727150e-02 -5.27946012e-02 -1.15542881e-01
 -1.93086972e-01  1.26465966e-01  9.01770405e-04  8.61173884e-02
  4.22681329e-02  5.91892370e-03 -3.31146133e-02 -4.57095884e-02
  2.50299607e-01  3.91698208e-03 -1.67603202e-02  1.40499461e-01
 -2.12289509e-02  1.40066155e-01  4.76268277e-02  5.58694526e-02
  4.33036012e-03 -5.47094347e-02 -4.43379439e-03 -1.17710382e-01
 -3.52040347e-02 -3.85428353e-02  1.58478156e-02  4.84926665e-02
  2.68926012e-03 -6.17499422e-02  8.98996111e-02  6.02097977e-02]
```

In [47]:

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

```
100%|████████████████████████████████████████████████████████████████████████████| 24155/24155
[00:16<00:00, 1492.88it/s]
```

In [48]:

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

```
100%|████████████████████████████████████████████████████████████████| 36052/36052
[00:22<00:00, 1575.06it/s]
```

# 1.4.4 encoding Text features: Essay-TFIDF W2Vec

In [49]:

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

In [50]:

```
tfidf_w2v_vectors_train = []; # the tfidf-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            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]))
print(tfidf_w2v_vectors_train[0])
```

```
100%|████████████████████████████████████████████████████████████████| 49041/49041 [05:
28<00:00, 149.30it/s]
```

```
49041
300
[-6.84672871e-02 -1.26054615e-01 -7.09081633e-02 -1.91463106e-01
  6.32615129e-02 -5.24169387e-02 -3.44281180e+00  1.63300096e-01
  8.58279717e-02 -1.27179637e-01  2.01308674e-01  1.60034891e-02
 -1.10808867e-01 -1.47321548e-01 -8.96137094e-02 -1.34737217e-01
 -1.00834070e-01 -1.25979124e-01  1.21675103e-01  1.49560737e-01
  1.99275143e-02  6.28080242e-02 -9.48723013e-02 -6.89930924e-03
 -9.71713177e-02  1.42238464e-02  4.68689933e-02 -1.54657897e-01
 -1.16597834e-01 -1.20062164e-01 -2.03603606e-01 -7.15663790e-02
  2.62403489e-02 -5.57915182e-02 -1.40994364e-01 -1.87230873e-02
  4.37058890e-02 -9.06354818e-02  1.33577135e-01 -8.74996951e-02
 -1.09839664e-02  9.28004129e-02  4.39318200e-02 -1.62534969e-01
 -5.12091334e-02 -5.21427640e-02  1.00652366e-01 -1.82101241e-01
 -1.56745817e-01  6.11073634e-02  6.44576438e-02  1.35466545e-01
 -2.89925318e-02 -6.04708127e-02  4.13553253e-02 -1.00632647e-01
  6.82864405e-02 -2.52207950e-03 -9.35790676e-02  4.62797838e-02
 -1.15577196e-03  1.11880569e-01  4.70715012e-02 -1.79723594e-01
 -1.01259149e-01  2.10982587e-01  1.11642136e-01  4.87169301e-02
  2.14559043e-01 -1.23641349e-01 -2.90260500e-01  2.75175348e-02
 -2.37113568e-02 -1.22933089e-01  6.73358697e-02 -2.81412134e-01
  8.80006023e-02 -5.36141407e-02  1.86131543e-01 -1.23881537e-01
  4.67219087e-02 -5.64298671e-01 -1.09768720e-01 -1.93196888e-01
  1.26059070e-01  1.35427537e-01  2.69003824e-02 -1.31236050e-01
  1.40777261e-01 -5.37157668e-02 -2.74373679e-03 -8.00607927e-02
  1.66333368e-02  6.31074416e-02 -1.59328565e-02 -2.33023476e-01
 -2.74031844e+00 -5.86049019e-02  1.02628717e-01  3.70052601e-02
```

```
 -3.24976130e-02  1.91230966e-01  1.65155535e-01  5.55977428e-02
-1.41981028e-01 -1.52303424e-02  1.20338207e-01 -1.70989801e-01
-9.98457448e-02  7.15161902e-02 -3.91864133e-03  1.57416637e-01
 1.11917488e-01  2.00129101e-01 -3.29900763e-02  2.37412459e-02
 5.51033899e-02 -4.78277596e-02  9.48436600e-02  7.77183246e-02
-1.85909621e-01 -3.18291285e-02 -3.61935167e-02 -1.29433601e-01
-1.15106581e-02 -9.30770845e-03 -6.69788938e-02 -8.25618515e-02
-1.23696398e-02  7.88797106e-02  1.11472340e-01  2.74019481e-02
-1.27624408e-02 -1.53128310e-01  3.08607085e-02  9.80613214e-02
 6.90222770e-02  2.51164934e-02  3.10020452e-01  2.32271263e-01
 3.87868557e-02  2.13188751e-02 -2.25521723e-02  4.01346891e-02
-4.41596126e-02 -6.27042525e-03  6.24715433e-02 -5.56046926e-02
 4.08768777e-01  1.55939433e-01 -8.88437666e-02 -4.22041464e-02
-3.00972233e-03 -1.85503449e-02 -3.59244349e-02 -1.12686893e-01
 7.93042899e-02  6.95315965e-02 -1.59756495e-01 -3.96502412e-02
 1.08224251e-01 -5.68083392e-02  5.78771151e-03 -2.56600813e-02
-9.83455477e-02  4.09632472e-02 -2.81906113e-02  1.47124962e-01
 1.49281656e-01 -8.03067420e-02 -2.64424495e-02  6.29797976e-02
-1.00709927e-01 -1.60810049e-02 -8.53325959e-02  3.03839098e-01
-1.22003209e-01 -1.65175289e-01 -9.00005568e-03  1.81308674e-02
 1.29907765e-01  6.29247820e-02 -1.19142617e-01 -1.70756022e-01
 9.97759280e-02 -2.14404142e-01  5.64748262e-02 -4.04756750e-02
 1.68660501e-01 -7.18098346e-02  2.84708674e-02 -5.02999161e-02
 1.42770840e-02 -1.34559786e-02  2.37321009e-02 -7.55841137e-02
 6.51121460e-02  1.09358737e-01  1.96251769e-01  3.13252766e-02
-1.11394288e-02 -1.31533657e-02  3.60516853e-03 -4.75406446e-02
 4.96080422e-02  1.61548607e-01  6.44507515e-02  3.22668666e-02
 1.49126804e-01  2.14354971e-02  7.78369652e-02 -5.25009146e-02
-9.99386807e-02 -2.99324973e-02 -9.74296039e-02  1.66470672e-02
-2.57013065e-02 -5.13221868e-02  3.62742596e-03  1.61699369e-01
-1.21392635e-01 -1.04983079e-01 -4.60739201e-02  5.39838051e-02
-2.67055143e+00  1.89292204e-02 -6.06988384e-02 -3.02009796e-02
 7.91566723e-03 -3.84404571e-02  9.46886827e-02  5.43227686e-02
 3.51910157e-02 -6.91096224e-02 -1.24559529e-01  1.16518302e-01
 2.78431724e-02 -7.44680978e-02 -1.39866498e-01  1.26412220e-01
-1.52161708e-01  1.34044415e-01 -1.08658871e-01  8.27447690e-03
 5.79859383e-02 -9.63250807e-02 -5.68080649e-02 -6.51253945e-02
-1.19115421e-01  2.74241022e-02 -8.17190092e-02 -7.34496874e-04
 5.07701340e-02 -3.10435667e-02  1.20813227e-02 -9.73730550e-03
 5.85570671e-02 -7.74782507e-02  1.53061264e-01  4.71772954e-02
 1.95304008e-01  3.84426460e-02 -4.45106984e-02  6.05754232e-02
 1.75232281e-01 -1.75824746e-01 -3.17975833e-01 -5.99350689e-02
 1.47423587e-01 -2.44517687e-02 -9.50426325e-02 -1.56503731e-01
-2.22481111e-01  1.94470508e-01 -1.06713247e-02  8.47595409e-02
 2.83632828e-02 -2.70410100e-02 -1.01717011e-01 -6.77483826e-02
 3.02951495e-01  2.72927008e-02 -2.69391232e-02  1.62829150e-01
-3.76784147e-02  1.56580684e-01  3.82787884e-02  2.35164373e-02
 4.47012163e-03 -8.01852803e-02  2.54845376e-02 -1.75930545e-01
 2.14619461e-02  2.51896111e-02  2.04584980e-02  4.92972336e-02
 5.27417043e-02 -1.17373876e-01  1.06799086e-01  5.23172084e-02]
```

In [51]:

```
tfidf_w2v_vectors_test = []; # the tfidf-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_test.append(vector)
```

```
100%|████████████████████████████████████████████████████████████████████████████| 36052/36052 [04:
00<00:00, 150.08it/s]
```

In [52]:

```
tfidf_w2v_vectors_cv = []; # the tfidf-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['essay'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
```

```
    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]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors_cv.append(vector)
```

```
100%|██████████████████████████████████████████████████████████████████████████| 24155/24155 [02:
40<00:00, 150.80it/s]
```

# 1.4.5 encoding Text features: Title-BOW

In [53]:

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

print("="*100)


vectorizer = CountVectorizer(min_df=10,ngram_range=(1,4), max_features=5000)
vectorizer.fit(X_train['project_title'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_project_title_bow = vectorizer.transform(X_train['project_title'].values)
X_cv_project_title_bow = vectorizer.transform(X_cv['project_title'].values)
X_test_project_title_bow = vectorizer.transform(X_test['project_title'].values)

print("After vectorizations")
print(X_train_project_title_bow.shape, y_train.shape)
print(X_cv_project_title_bow.shape, y_cv.shape)
print(X_test_project_title_bow.shape, y_test.shape)
print("="*100)



print("NOTE: THE NUMBER OF COLUMNS IN EACH OF THE VECTOR WONT BE SAME")
```

```
(49041, 23) (49041,)
(24155, 23) (24155,)
(36052, 23) (36052,)
====================================================================================================

After vectorizations
(49041, 5000) (49041,)
(24155, 5000) (24155,)
(36052, 5000) (36052,)
====================================================================================================

NOTE: THE NUMBER OF COLUMNS IN EACH OF THE VECTOR WONT BE SAME
```

# 1.4.6 encoding Text features: Tfidf

In [54]:

```
from sklearn.feature_extraction.text import TfidfVectorizer

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

print("="*100)


vectorizer_title_Tfidf = TfidfVectorizer(min_df=10, max_features=5000)
```

```
vectorizer_title_Tfidf = TfidfVectorizer(min_df=10, max_features=5000)
vectorizer_title_Tfidf.fit(X_train['project_title'].values) # fit has to happen only on train data

# we use the fitted TfidfVectorizer to convert the text to vector
X_train_project_title_Tfidf = vectorizer_title_Tfidf.transform(X_train['project_title'].values)
X_cv_project_title_Tfidf = vectorizer_title_Tfidf.transform(X_cv['project_title'].values)
X_test_project_title_Tfidf = vectorizer_title_Tfidf.transform(X_test['project_title'].values)

print("After vectorizations")
print(X_train_project_title_Tfidf.shape, y_train.shape)
print(X_cv_project_title_Tfidf.shape, y_cv.shape)
print(X_test_project_title_Tfidf.shape, y_test.shape)
print("="*100)



print("NOTE: THE NUMBER OF COLUMNS IN EACH OF THE VECTOR WONT BE SAME")
```

```
(49041, 23) (49041,)
(24155, 23) (24155,)
(36052, 23) (36052,)
====================================================================================================

After vectorizations
(49041, 2112) (49041,)
(24155, 2112) (24155,)
(36052, 2112) (36052,)
====================================================================================================

NOTE: THE NUMBER OF COLUMNS IN EACH OF THE VECTOR WONT BE SAME
```

# 1.4.7 encoding Text features: Title-Avg W2Vec

In [55]:

```
avg_w2v_project_title_vectors_train = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(X_train['project_title'].values): # 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_project_title_vectors_train.append(vector)

print(len(avg_w2v_project_title_vectors_train))
print(len(avg_w2v_project_title_vectors_train[0]))
print(avg_w2v_project_title_vectors_train[0])
```

```
100%|████████████████████████████████████████████████████████████████████████████| 49041/49041
[00:02<00:00, 17877.11it/s]

49041
300
[-1.7614215e-02  4.1353550e-01 -1.4185000e-02  2.5422600e-01
 -2.5891500e-01 -1.4724155e-01 -3.9256000e+00  6.4819000e-01
  1.4414250e-01 -5.0458000e-01  1.1050000e-03 -3.1982500e-01
 -3.5685500e-01 -1.7586500e-01  2.8473500e-01 -3.5333000e-01
 -1.0059000e-01  2.2769500e-01 -2.7086500e-02  1.4461300e-01
  2.4896650e-01 -1.5500500e-01 -2.3905000e-02 -1.8937150e-01
 -2.2266323e-01 -3.6061850e-01 -1.6995500e-01 -1.8016500e-01
 -9.3865000e-02  7.4765000e-02 -1.4045750e-01 -1.6811500e-01
  7.6218000e-02  1.1300950e-01 -3.3151300e-02  2.7876000e-01
  4.6675000e-02 -1.8467500e-01  2.5315000e-02 -1.9969950e-01
 -3.4567000e-01 -1.2675000e-01  1.2430000e-02 -4.5979500e-01
 -4.2620500e-02 -3.9868500e-01  1.9268500e-02 -7.0350000e-03
 -2.1759600e-01  2.6683500e-01 -1.3123000e-02  3.9906950e-01
 -5.6853000e-01 -4.0831000e-02  1.8581400e-01 -2.0338500e-01
  1.8546260e-01 -1.2643950e-01  2.7400000e-01  2.6371200e-01
 -9.9646500e-02  2.4919200e-01 -1.3615000e-02  1.4704500e-01
```

```
-2.2670000e-02 -5.9560000e-02 -1.9130000e-02  3.9276000e-01
 1.4870555e-01 -2.5742000e-01 -9.2570000e-02  2.2301500e-02
-4.3505000e-01  7.3899500e-02 -6.5329500e-02 -3.2726300e-01
 3.2461500e-01  1.0360000e-03 -2.2227000e-01 -2.4818000e-01
-3.5135000e-01 -5.7140500e-01 -9.7590000e-02 -2.7189685e-01
-9.6775000e-02  2.7795000e-02 -5.4749000e-01  1.0340750e-01
-1.3315000e-02  4.3702500e-02  2.9985500e-01 -2.1908100e-01
-1.8219800e-01  3.1173000e-01 -4.5820000e-02 -5.7039500e-01
-2.6310000e+00  1.8915000e-02  8.9969000e-02  1.0117500e-01
 1.2690550e-01 -1.2190500e-01  2.6428000e-01 -2.1221000e-01
-4.1645450e-02 -2.2501500e-01  4.6250000e-02  1.2609740e-01
-2.2996000e-01 -2.2659600e-01 -6.7940000e-02 -1.8287000e-01
 3.5286500e-01  3.4285000e-02 -2.2578700e-01 -3.3058500e-02
 7.5865500e-01 -1.1065350e-01  1.0208000e-01 -5.0882000e-01
-7.2183000e-01  2.7957950e-01  1.7704000e-02 -3.9317500e-01
 6.6910000e-02 -1.5628510e-01 -1.2465000e-01  1.5442000e-02
-2.8304000e-02  2.1781305e-01 -1.4817500e-01  4.5531000e-01
 2.0945000e-01 -1.1852200e-01  4.9100000e-02  7.6121500e-02
 1.4529550e-01 -3.8449000e-02  2.8756500e-01  6.7941500e-01
-3.5150000e-04  1.7594000e-01  3.9059500e-01 -1.7772000e-01
-1.0894000e-01  2.7297900e-02  4.1431000e-01 -3.5924500e-01
 6.9263500e-01  5.5159500e-01 -5.8140000e-02 -1.4537500e-01
 1.7508505e-01  6.1700000e-03  4.0179500e-01 -4.0816000e-01
-9.7450000e-02  1.3130450e-02 -1.8283000e-01 -2.6466500e-01
 4.2035000e-01  5.8260000e-02 -1.7050500e-01  4.0125000e-02
-1.4662430e-01 -7.3365000e-02 -7.1234500e-02 -1.1982500e-01
 3.1214500e-02 -7.1207900e-02 -2.3151500e-01 -2.9659500e-01
 2.1195000e-01 -2.2474500e-01 -8.5108500e-02 -4.8925000e-02
-2.3860725e-01 -1.1372550e-01  6.3765000e-02  9.8262500e-02
 2.8711500e-01  3.6239800e-01 -1.5326880e-01  2.1327050e-01
-1.1212050e-01 -2.2206650e-01 -1.5062715e-01  1.2448500e-01
 1.8462000e-01 -1.4449000e-01  7.7735000e-02 -4.2715000e-01
-1.4471500e-01  7.1521500e-02  1.1883600e-01  4.0022050e-02
-2.9691500e-02  9.3706500e-02  1.1581700e-01 -1.0108550e-01
 2.7232300e-01 -1.9871445e-02 -2.6795000e-02 -2.0376000e-01
 2.3483000e-01  1.4090000e-01 -1.0482150e-01 -2.0298000e-01
 1.6226300e-01  7.0965000e-02  1.1765000e-02 -2.1448450e-01
-2.1076000e-01  2.2274500e-01 -3.2340000e-02 -3.2911500e-01
-4.1186500e-01 -7.5100000e-03 -1.5189750e-01  8.7260500e-02
-1.1192500e-01  1.8685500e-01  5.6490000e-02  1.9922200e-01
-3.6314000e+00  8.4930000e-02 -2.8582500e-01 -1.8341100e-01
 1.1689500e-01 -7.0247000e-02  3.5591000e-01  2.7824250e-01
-1.4322950e-01 -1.2196490e-01  1.4183950e-01  1.8657100e-01
-1.7971200e-01  9.9950000e-02 -2.6499500e-01 -3.7640500e-01
-2.3522765e-01  1.7056400e-01 -3.9675000e-01  2.6101500e-01
-2.2069000e-01 -2.0400000e-03  1.0106500e-02 -1.1416335e-01
-3.0274500e-01  4.6183500e-01 -2.4435000e-02 -1.8769200e-01
 1.0643750e-01  6.0653970e-02  8.6297300e-02  1.4120900e-01
-5.2490000e-01 -4.8409000e-01  7.5230000e-02 -5.2465000e-02
 4.6150000e-02 -2.4811500e-01  1.8818600e-01  2.9123000e-01
-4.1595000e-02 -4.3035000e-02 -4.4752000e-01 -4.6059000e-01
 3.4821000e-01 -3.8587500e-02  5.1892000e-01 -1.7719600e-01
-1.2520500e-01 -9.7050000e-03  3.1443500e-01 -1.1628300e-01
 1.5553350e-01  4.3530000e-02  1.2609500e-01 -3.0214000e-01
 5.6143500e-01 -6.7620000e-02 -1.2555700e-01 -1.2911700e-01
 4.4103200e-01 -1.4802500e-01  3.8171500e-01  5.2673000e-01
-2.6362500e-01 -2.0410000e-02  1.7817350e-01  1.8785500e-01
-2.6454000e-01  8.2715000e-02 -1.2020000e-01 -2.8413000e-01
-2.4539485e-01  7.9815500e-02 -8.8625000e-02 -2.1040000e-02]
```

In [56]:

```
avg_w2v_project_title_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this li
st
for sentence in tqdm(X_cv['project_title'].values): # 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_project_title_vectors_cv.append(vector)
```

```
100%|████████████████████████████████████████████████████████████████| 24155/24155
[00:00<00:00, 29123.90it/s]
```

In [57]:

```
avg_w2v_project_title_vectors_test = []; # the avg-w2v for each sentence/review is stored in this
list
for sentence in tqdm(X_test['project_title'].values): # 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_project_title_vectors_test.append(vector)
```

```
100%|████████████████████████████████████████████████████████████████| 36052/36052
[00:01<00:00, 26677.62it/s]
```

# 1.4.8 encoding Text features: Title-Tfidf W2Vec

In [58]:

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

In [59]:

```
tfidf_w2v_project_title_vectors_train = []; # the tfidf-w2v for each sentence/review is stored in
this list
for sentence in tqdm(X_train['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_project_title_vectors_train.append(vector)

print(len(tfidf_w2v_project_title_vectors_train))
print(len(tfidf_w2v_project_title_vectors_train[0]))
print(tfidf_w2v_project_title_vectors_train[0])
```

```
100%|████████████████████████████████████████████████████████████████| 49041/49041
[00:01<00:00, 47433.39it/s]
```

```
49041
300
[ 6.0657e-04  4.8631e-02  4.8969e-01  4.2777e-01 -3.8610e-01 -8.4231e-03
 -3.6027e+00  4.7811e-01  4.7945e-02 -3.1859e-01 -2.1335e-01 -5.1531e-01
 -1.7142e-01 -2.0035e-01  7.0538e-01 -1.7186e-01 -5.4713e-01  6.8465e-01
 -4.0384e-02  2.4141e-01  5.3936e-01  1.0057e-01 -1.4953e-01 -2.8165e-01
 -4.4468e-01 -6.9436e-01 -1.0518e-01 -3.0014e-01  1.2637e-01  7.1193e-01
 -2.0152e-01 -1.1507e-01  5.3484e-02  1.3611e-01  2.3964e-03  2.4965e-01
 -2.6389e-01 -6.4683e-01 -1.6104e-01 -3.7846e-01 -2.0946e-01 -1.0369e-01
  2.2179e-01 -7.2445e-01 -5.5569e-02 -5.1117e-01 -2.0982e-02  1.4127e-01
 -3.9321e-01  2.2062e-01  8.2094e-02  5.3219e-02 -8.6973e-01 -1.7208e-01
  4.3798e-02 -5.6050e-02  3.6899e-01 -9.2699e-02  1.9369e-01  5.9429e-01
 -1.0439e-01  5.2525e-01  4.1624e-01  1.5565e-01 -1.9486e-01 -4.6395e-01
 -1.6372e-01  3.2242e-01  3.0328e-01 -1.0038e-01 -3.1126e-01 -9.1417e-02
```

```
 -4.3295e-01  5.1531e-02  5.9131e-02  9.9654e-02  2.9146e-01  1.9551e-02
 -2.2547e-01 -3.1254e-01 -4.3268e-01 -5.9699e-01  2.5903e-01  6.1463e-03
  2.3724e-01  1.7422e-01 -6.2129e-01  3.0664e-01  1.1598e-01  1.6893e-01
  2.6485e-01 -2.3562e-02 -2.8476e-01  3.2007e-01  3.3411e-01 -7.9123e-01
 -2.5263e+00  7.7887e-01 -3.6362e-02 -2.2488e-01  2.1838e-01 -1.8917e-01
 -1.7813e-01 -1.2703e-01 -9.1610e-02 -2.5722e-01 -2.3141e-01  2.4344e-01
 -2.9977e-01 -3.9411e-01 -3.1091e-01 -7.1663e-01  7.1927e-01  4.0314e-01
 -3.9980e-01 -8.1807e-02  4.8831e-01 -2.6558e-01 -2.4490e-01 -7.2728e-01
 -6.5682e-01  6.1916e-01 -8.2432e-02 -6.6695e-01 -7.1830e-02 -6.0302e-03
 -1.4047e-01  1.0774e-01 -2.9400e-02  4.3884e-01 -1.0434e-01  7.7235e-01
  2.6651e-01 -1.5147e-01 -1.5951e-01  1.1666e-01  3.3501e-02  5.5742e-02
 -7.0720e-02  8.1983e-01 -9.4083e-02  1.7576e-01  3.2675e-01 -5.2920e-02
  2.9963e-01  6.2303e-02  3.0784e-01 -4.8853e-01  8.7214e-01  4.8469e-01
  1.5472e-01 -4.7794e-01  3.5701e-03  2.8510e-01  4.5573e-01 -4.9919e-01
 -3.2528e-01  5.0259e-03 -1.0745e-01 -3.8380e-01  1.8128e-01  3.1121e-01
 -1.6429e-01  2.5904e-01 -2.0586e-03 -4.2063e-01 -6.7520e-02  1.1249e-01
 -6.2841e-02  9.0842e-03 -6.2864e-01 -2.3783e-01  1.6221e-01  2.2003e-01
 -1.3259e-01 -3.4341e-01 -3.8345e-03 -3.1361e-02  2.9267e-01  1.1335e-02
 -1.6943e-01  6.2525e-01  1.8924e-03  3.4430e-01 -7.7261e-02  9.0077e-02
 -2.9513e-01 -3.2870e-01  2.3480e-02 -1.2653e-01  9.3243e-02 -5.8030e-01
 -1.8007e-01 -3.6697e-02 -9.1138e-02  8.0947e-02 -1.3167e-01  6.9933e-02
 -1.9786e-02 -2.6451e-01  9.5616e-02 -3.8800e-02 -2.3523e-01 -1.1224e-01
  2.9692e-01 -1.3358e-01 -1.8443e-02 -5.5351e-01 -7.1414e-02  3.0206e-01
 -4.0702e-02 -9.9249e-02 -6.2182e-01  1.3056e-01  3.7276e-01 -5.1884e-01
 -7.2358e-01  1.9346e-01 -7.7715e-02  2.2269e-01 -4.3023e-01 -1.3878e-01
  4.5398e-01  4.0267e-01 -3.7990e+00  2.9864e-01 -1.0014e-01  5.3288e-02
  2.7437e-01 -1.8284e-01  5.9567e-01  5.9842e-01 -3.2466e-01  3.5302e-03
  3.7720e-01 -7.2068e-02 -3.1471e-01 -1.2439e-01 -3.9985e-01 -3.6131e-01
 -9.4553e-03  4.1125e-01 -4.4127e-01  1.3517e-01 -1.8269e-01 -4.7905e-01
 -5.6259e-02 -2.3687e-01 -2.0079e-01  2.6212e-01  1.8157e-01 -3.5366e-01
  2.3476e-01  1.2185e-01  1.6606e-01  3.5740e-01 -4.5782e-01 -3.5614e-01
  2.9046e-01 -2.9956e-01  2.6952e-01  1.4340e-02 -2.6618e-02  3.1496e-01
  2.1012e-01  1.0300e-01 -5.1883e-01 -5.0328e-01  1.4767e-01 -1.4743e-01
  2.3220e-01  2.9708e-02  1.7360e-01  3.9443e-01  1.9012e-01  5.9954e-02
  4.0225e-01 -1.3333e-01  4.3591e-01 -1.9416e-01  4.9440e-01 -3.5507e-01
 -1.7199e-01 -2.7750e-01  2.6634e-02 -1.6227e-01  1.6202e-01  3.7628e-01
 -1.9416e-01  2.1528e-01  2.7162e-01  2.7112e-01 -2.7549e-01  2.8545e-01
  1.4925e-01 -2.8046e-01 -4.9446e-01  1.3731e-01  1.3866e-01  3.1400e-01]
```

In [60]:

```
tfidf_w2v_project_title_vectors_test = []; # the tfidf-w2v for each sentence/review is stored in t
his list
for sentence in tqdm(X_test['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_project_title_vectors_test.append(vector)
```

```
100%|██████████████████████████████████████████████████████████████████████████| 36052/36052
[00:00<00:00, 46948.79it/s]
```

In [61]:

```
tfidf_w2v_project_title_vectors_cv = []; # the tfidf-w2v for each sentence/review is stored in thi
s list
for sentence in tqdm(X_cv['project_title'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word]
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            vector += (vec * tf_idf)
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_project_title_vectors_cv.append(vector)
```

```
    tfidf_w2v_project_title_vectors_cv.append(vector)
```

```
100%|████████████████████████████████████████████████████████████████| 24155/24155 [00:00<00:00, 45426.93it/s]
```

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

## 1.5.1 encoding categorical features: Project_subject_categories-ohe

In [62]:

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

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

In [63]:

```
vectorizer_cat=CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
```

In [64]:

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_project_subject_categories_ohe = vectorizer.transform(X_train['project_subject_categories'].values)
X_cv_project_subject_categories_ohe = vectorizer.transform(X_cv['project_subject_categories'].values)
X_test_project_subject_categories_ohe = vectorizer.transform(X_test['project_subject_categories'].values)

print("After vectorizations")
print(X_train_project_subject_categories_ohe.shape, y_train.shape)
print(X_cv_project_subject_categories_ohe.shape, y_cv.shape)
print(X_test_project_subject_categories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(49041, 51) (49041,)
(24155, 51) (24155,)
(36052, 51) (36052,)
['appliedlearning', 'appliedlearning_health_sports', 'appliedlearning_history_civics',
'appliedlearning_literacy_language', 'appliedlearning_math_science', 'appliedlearning_music_arts',
'appliedlearning_specialneeds', 'appliedlearning_warmth_care_hunger', 'health_sports',
'health_sports_appliedlearning', 'health_sports_history_civics',
'health_sports_literacy_language', 'health_sports_math_science', 'health_sports_music_arts',
'health_sports_specialneeds', 'health_sports_warmth_care_hunger', 'history_civics',
'history_civics_appliedlearning', 'history_civics_health_sports',
'history_civics_literacy_language', 'history_civics_math_science', 'history_civics_music_arts', 'history_civics_specialneeds', 'history_civics_warmth_care_hunger', 'literacy_language',
'literacy_language_appliedlearning', 'literacy_language_health_sports',
'literacy_language_history_civics', 'literacy_language_math_science',
'literacy_language_music_arts', 'literacy_language_specialneeds',
'literacy_language_warmth_care_hunger', 'math_science', 'math_science_appliedlearning',
'math_science_health_sports', 'math_science_history_civics', 'math_science_literacy_language',
'math_science_music_arts', 'math_science_specialneeds', 'math_science_warmth_care_hunger',
'music_arts', 'music_arts_appliedlearning', 'music_arts_health_sports',
'music_arts_history_civics', 'music_arts_specialneeds', 'music_arts_warmth_care_hunger',
'specialneeds', 'specialneeds_health_sports', 'specialneeds_music_arts',
'specialneeds_warmth_care_hunger', 'warmth_care_hunger']
====================================================================================================
```

## 1.5.2 encoding categorical features: Project_subject_subcategories-ohe

In [65]:

```
my_counter = Counter()
for word in project_data['project_subject_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 t: t[1]))
```

In [66]:

```
vectorizer_subcat=CountVectorizer(vocabulary=list(sub_cat_dict.keys()), lowercase=False, binary=True)
```

In [67]:

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

# we use the fitted CountVectorizer to convert the text to vector
X_train_project_subject_subcategories_ohe =
vectorizer.transform(X_train['project_subject_subcategories'].values)
X_cv_project_subject_subcategories_ohe = vectorizer.transform(X_cv['project_subject_subcategories'
].values)
X_test_project_subject_subcategories_ohe =
vectorizer.transform(X_test['project_subject_subcategories'].values)

print("After vectorizations")
print(X_train_project_subject_subcategories_ohe.shape, y_train.shape)
print(X_cv_project_subject_subcategories_ohe.shape, y_cv.shape)
print(X_test_project_subject_subcategories_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(49041, 387) (49041,)
(24155, 387) (24155,)
(36052, 387) (36052,)
['appliedsciences', 'appliedsciences_charactereducation', 'appliedsciences_civics_government',
'appliedsciences_college_careerprep', 'appliedsciences_communityservice',
'appliedsciences_earlydevelopment', 'appliedsciences_economics',
'appliedsciences_environmentalscience', 'appliedsciences_esl', 'appliedsciences_extracurricular',
'appliedsciences_financialliteracy', 'appliedsciences_foreignlanguages',
'appliedsciences_gym_fitness', 'appliedsciences_health_lifescience',
'appliedsciences_health_wellness', 'appliedsciences_history_geography',
'appliedsciences_literacy', 'appliedsciences_literature_writing', 'appliedsciences_mathematics', '
appliedsciences_music', 'appliedsciences_nutritioneducation', 'appliedsciences_other',
'appliedsciences_parentinvolvement', 'appliedsciences_performingarts',
'appliedsciences_socialsciences', 'appliedsciences_specialneeds', 'appliedsciences_teamsports', 'a
ppliedsciences_visualarts', 'appliedsciences_warmth_care_hunger', 'charactereducation',
'charactereducation_civics_government', 'charactereducation_college_careerprep',
'charactereducation_communityservice', 'charactereducation_earlydevelopment',
'charactereducation_economics', 'charactereducation_environmentalscience',
'charactereducation_esl', 'charactereducation_extracurricular',
'charactereducation_financialliteracy', 'charactereducation_foreignlanguages',
'charactereducation_gym_fitness', 'charactereducation_health_lifescience',
'charactereducation_health_wellness', 'charactereducation_history_geography',
'charactereducation_literacy', 'charactereducation_literature_writing',
'charactereducation_mathematics', 'charactereducation_music', 'charactereducation_other',
'charactereducation_parentinvolvement', 'charactereducation_performingarts',
'charactereducation_socialsciences', 'charactereducation_specialneeds',
'charactereducation_teamsports', 'charactereducation_visualarts',
'charactereducation_warmth_care_hunger', 'civics_government',
'civics_government_college_careerprep', 'civics_government_communityservice',
'civics_government_economics', 'civics_government_environmentalscience', 'civics_government_esl',
```

```
'civics_government_extracurricular', 'civics_government_financialliteracy',
'civics_government_foreignlanguages', 'civics_government_health_lifescience',
'civics_government_health_wellness', 'civics_government_history_geography',
'civics_government_literacy', 'civics_government_literature_writing',
'civics_government_mathematics', 'civics_government_nutritioneducation',
'civics_government_performingarts', 'civics_government_socialsciences',
'civics_government_specialneeds', 'civics_government_teamsports', 'civics_government_visualarts',
'college_careerprep', 'college_careerprep_communityservice',
'college_careerprep_earlydevelopment', 'college_careerprep_economics',
'college_careerprep_environmentalscience', 'college_careerprep_esl',
'college_careerprep_extracurricular', 'college_careerprep_financialliteracy',
'college_careerprep_foreignlanguages', 'college_careerprep_gym_fitness',
'college_careerprep_health_lifescience', 'college_careerprep_health_wellness',
'college_careerprep_history_geography', 'college_careerprep_literacy',
'college_careerprep_literature_writing', 'college_careerprep_mathematics',
'college_careerprep_music', 'college_careerprep_nutritioneducation', 'college_careerprep_other', '
college_careerprep_parentinvolvement', 'college_careerprep_performingarts',
'college_careerprep_socialsciences', 'college_careerprep_specialneeds',
'college_careerprep_teamsports', 'college_careerprep_visualarts', 'communityservice',
'communityservice_earlydevelopment', 'communityservice_economics',
'communityservice_environmentalscience', 'communityservice_esl',
'communityservice_extracurricular', 'communityservice_gym_fitness',
'communityservice_health_lifescience', 'communityservice_health_wellness',
'communityservice_history_geography', 'communityservice_literacy',
'communityservice_literature_writing', 'communityservice_mathematics',
'communityservice_nutritioneducation', 'communityservice_other',
'communityservice_parentinvolvement', 'communityservice_performingarts',
'communityservice_socialsciences', 'communityservice_specialneeds', 'communityservice_visualarts',
'earlydevelopment', 'earlydevelopment_economics', 'earlydevelopment_environmentalscience',
'earlydevelopment_extracurricular', 'earlydevelopment_financialliteracy',
'earlydevelopment_foreignlanguages', 'earlydevelopment_gym_fitness',
'earlydevelopment_health_lifescience', 'earlydevelopment_health_wellness',
'earlydevelopment_history_geography', 'earlydevelopment_literacy',
'earlydevelopment_literature_writing', 'earlydevelopment_mathematics', 'earlydevelopment_music', '
earlydevelopment_nutritioneducation', 'earlydevelopment_other',
'earlydevelopment_parentinvolvement', 'earlydevelopment_performingarts',
'earlydevelopment_socialsciences', 'earlydevelopment_specialneeds', 'earlydevelopment_teamsports',
'earlydevelopment_visualarts', 'earlydevelopment_warmth_care_hunger', 'economics',
'economics_environmentalscience', 'economics_financialliteracy', 'economics_foreignlanguages', 'ec
onomics_history_geography', 'economics_literacy', 'economics_mathematics',
'economics_nutritioneducation', 'economics_socialsciences', 'economics_specialneeds',
'economics_visualarts', 'environmentalscience', 'environmentalscience_extracurricular',
'environmentalscience_financialliteracy', 'environmentalscience_foreignlanguages',
'environmentalscience_health_lifescience', 'environmentalscience_health_wellness',
'environmentalscience_history_geography', 'environmentalscience_literacy',
'environmentalscience_literature_writing', 'environmentalscience_mathematics',
'environmentalscience_music', 'environmentalscience_nutritioneducation',
'environmentalscience_other', 'environmentalscience_parentinvolvement',
'environmentalscience_performingarts', 'environmentalscience_socialsciences',
'environmentalscience_specialneeds', 'environmentalscience_teamsports',
'environmentalscience_visualarts', 'environmentalscience_warmth_care_hunger', 'esl',
'esl_earlydevelopment', 'esl_economics', 'esl_environmentalscience', 'esl_extracurricular',
'esl_financialliteracy', 'esl_foreignlanguages', 'esl_gym_fitness', 'esl_health_lifescience',
'esl_health_wellness', 'esl_history_geography', 'esl_literacy', 'esl_literature_writing',
'esl_mathematics', 'esl_music', 'esl_nutritioneducation', 'esl_other', 'esl_parentinvolvement', 'e
sl_performingarts', 'esl_socialsciences', 'esl_specialneeds', 'esl_teamsports', 'esl_visualarts',
'extracurricular', 'extracurricular_financialliteracy', 'extracurricular_foreignlanguages',
'extracurricular_gym_fitness', 'extracurricular_health_lifescience',
'extracurricular_health_wellness', 'extracurricular_history_geography',
'extracurricular_literacy', 'extracurricular_literature_writing', 'extracurricular_mathematics', '
extracurricular_music', 'extracurricular_nutritioneducation', 'extracurricular_other',
'extracurricular_parentinvolvement', 'extracurricular_performingarts',
'extracurricular_socialsciences', 'extracurricular_specialneeds', 'extracurricular_teamsports', 'e
xtracurricular_visualarts', 'financialliteracy', 'financialliteracy_foreignlanguages',
'financialliteracy_health_lifescience', 'financialliteracy_health_wellness',
'financialliteracy_history_geography', 'financialliteracy_literacy',
'financialliteracy_literature_writing', 'financialliteracy_mathematics',
'financialliteracy_other', 'financialliteracy_parentinvolvement',
'financialliteracy_socialsciences', 'financialliteracy_specialneeds',
'financialliteracy_visualarts', 'foreignlanguages', 'foreignlanguages_gym_fitness',
'foreignlanguages_health_lifescience', 'foreignlanguages_health_wellness',
'foreignlanguages_history_geography', 'foreignlanguages_literacy',
'foreignlanguages_literature_writing', 'foreignlanguages_mathematics', 'foreignlanguages_music', '
foreignlanguages_other', 'foreignlanguages_socialsciences', 'foreignlanguages_specialneeds',
'foreignlanguages_visualarts', 'gym_fitness', 'gym_fitness_health_lifescience',
'gym_fitness_health_wellness', 'gym_fitness_history_geography', 'gym_fitness_literacy',
'gym_fitness_literature_writing', 'gym_fitness_mathematics', 'gym_fitness_music',
```

```
'gym_fitness_nutritioneducation', 'gym_fitness_other', 'gym_fitness_parentinvolvement',
'gym_fitness_performingarts', 'gym_fitness_specialneeds', 'gym_fitness_teamsports',
'gym_fitness_visualarts', 'gym_fitness_warmth_care_hunger', 'health_lifescience',
'health_lifescience_health_wellness', 'health_lifescience_history_geography',
'health_lifescience_literacy', 'health_lifescience_literature_writing',
'health_lifescience_mathematics', 'health_lifescience_music',
'health_lifescience_nutritioneducation', 'health_lifescience_other',
'health_lifescience_parentinvolvement', 'health_lifescience_performingarts',
'health_lifescience_socialsciences', 'health_lifescience_specialneeds',
'health_lifescience_teamsports', 'health_lifescience_visualarts',
'health_lifescience_warmth_care_hunger', 'health_wellness', 'health_wellness_history_geography', '
health_wellness_literacy', 'health_wellness_literature_writing', 'health_wellness_mathematics', 'h
ealth_wellness_music', 'health_wellness_nutritioneducation', 'health_wellness_other',
'health_wellness_parentinvolvement', 'health_wellness_performingarts',
'health_wellness_socialsciences', 'health_wellness_specialneeds', 'health_wellness_teamsports', 'h
ealth_wellness_visualarts', 'health_wellness_warmth_care_hunger', 'history_geography',
'history_geography_literacy', 'history_geography_literature_writing',
'history_geography_mathematics', 'history_geography_music', 'history_geography_other',
'history_geography_parentinvolvement', 'history_geography_performingarts',
'history_geography_socialsciences', 'history_geography_specialneeds',
'history_geography_teamsports', 'history_geography_visualarts',
'history_geography_warmth_care_hunger', 'literacy', 'literacy_literature_writing',
'literacy_mathematics', 'literacy_music', 'literacy_nutritioneducation', 'literacy_other',
'literacy_parentinvolvement', 'literacy_performingarts', 'literacy_socialsciences',
'literacy_specialneeds', 'literacy_teamsports', 'literacy_visualarts',
'literacy_warmth_care_hunger', 'literature_writing', 'literature_writing_mathematics',
'literature_writing_music', 'literature_writing_nutritioneducation', 'literature_writing_other', '
literature_writing_parentinvolvement', 'literature_writing_performingarts',
'literature_writing_socialsciences', 'literature_writing_specialneeds',
'literature_writing_teamsports', 'literature_writing_visualarts',
'literature_writing_warmth_care_hunger', 'mathematics', 'mathematics_music',
'mathematics_nutritioneducation', 'mathematics_other', 'mathematics_parentinvolvement',
'mathematics_performingarts', 'mathematics_socialsciences', 'mathematics_specialneeds',
'mathematics_teamsports', 'mathematics_visualarts', 'mathematics_warmth_care_hunger', 'music', 'mu
sic_other', 'music_parentinvolvement', 'music_performingarts', 'music_socialsciences',
'music_specialneeds', 'music_teamsports', 'music_visualarts', 'nutritioneducation',
'nutritioneducation_other', 'nutritioneducation_socialsciences',
'nutritioneducation_specialneeds', 'nutritioneducation_teamsports',
'nutritioneducation_visualarts', 'nutritioneducation_warmth_care_hunger', 'other',
'other_parentinvolvement', 'other_performingarts', 'other_socialsciences', 'other_specialneeds', '
other_teamsports', 'other_visualarts', 'other_warmth_care_hunger', 'parentinvolvement',
'parentinvolvement_performingarts', 'parentinvolvement_socialsciences',
'parentinvolvement_specialneeds', 'parentinvolvement_teamsports', 'parentinvolvement_visualarts',
'performingarts', 'performingarts_socialsciences', 'performingarts_specialneeds',
'performingarts_teamsports', 'performingarts_visualarts', 'socialsciences',
'socialsciences_specialneeds', 'socialsciences_teamsports', 'socialsciences_visualarts',
'specialneeds', 'specialneeds_teamsports', 'specialneeds_visualarts',
'specialneeds_warmth_care_hunger', 'teamsports', 'teamsports_visualarts', 'visualarts',
'visualarts_warmth_care_hunger', 'warmth_care_hunger']
==========================================================================================
```

## 1.5.3 encoding categorical features: School State-ohe

In [68]:

```
for state in project_data['school_state'].values:
    my_counter.update(state.split())


school_state_cat_dict = dict(my_counter)
sorted_school_state_cat_dict = dict(sorted(school_state_cat_dict.items(), key=lambda t: t[1]))
```

In [69]:

```
vectorizer_state=CountVectorizer(vocabulary=list(sorted_school_state_cat_dict.keys()), lowercase=F
alse, binary=True)
```

In [70]:

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

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

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(49041, 51) (49041,)
(24155, 51) (24155,)
(36052, 51) (36052,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'k
s', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm',
'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv
', 'wy']
====================================================================================================
```

## 1.5.4 encoding categorical features: teacher_prefix-ohe

In [71]:

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

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

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

```
After vectorizations
(49041, 5) (49041,)
(24155, 5) (24155,)
(36052, 5) (36052,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
====================================================================================================
```

In [72]:

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


teacher_prefix_dict = dict(my_counter)
sorted_teacher_prefix_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda t: t[1]))
```

In [73]:

```
vectorizer_teacher=CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict.keys()), lowercase=F
alse, binary=True)
```

## 1.5.5 encoding categorical features: project_grade_category-ohe

In [74]:

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

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

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

```
After vectorizations
(49041, 4) (49041,)
(24155, 4) (24155,)
(36052, 4) (36052,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
====================================================================================================
```

In [75]:

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

project_grade_cat_dict = dict(my_counter)
sorted_project_grade_cat_dict = dict(sorted(project_grade_cat_dict.items(), key=lambda t: t[1]))
```

In [76]:

```
vectorizer_grade=CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()), lowercase=
False, binary=True)
```

## 1.5.6 encoding numerical features: Price

In [77]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(1,-1))

X_train_price_norm = normalizer.transform(X_train['price'].values.reshape(-1,1))
X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))



print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

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

==========

## 1.5.7 encoding numerical features: quantity

In [78]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['quantity'].values.reshape(1,-1))

X_train_quantity_norm = normalizer.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_quantity_norm = normalizer.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_quantity_norm = normalizer.transform(X_test['quantity'].values.reshape(-1,1))



print("After vectorizations")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_cv_quantity_norm.shape, y_cv.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

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

## 1.5.9 encoding numerical features: Projects_previously_proposed_by_teacher

In [79]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

X_train_num_prev_projects_norm =
normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_cv_num_prev_projects_norm =
normalizer.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))
X_test_num_prev_projects_norm =
normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))



print("After vectorizations")
print(X_train_num_prev_projects_norm.shape, y_train.shape)
print(X_cv_num_prev_projects_norm.shape, y_cv.shape)
print(X_test_num_prev_projects_norm.shape, y_test.shape)
print("="*100)
```

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

```
==========================================================================================================
```

## 1.5.9 encoding numerical features: title_count

In [80]:

```
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['title_number_words'].values.reshape(1,-1))

X_train_title_number_words_norm =
normalizer.transform(X_train['title_number_words'].values.reshape(-1,1))
X_cv_title_number_words_norm = normalizer.transform(X_cv['title_number_words'].values.reshape(-1,1)
)
X_test_title_number_words_norm = normalizer.transform(X_test['title_number_words'].values.reshape(
-1,1))



print("After vectorizations")
print(X_train_title_number_words_norm.shape, y_train.shape)
print(X_cv_title_number_words_norm.shape, y_cv.shape)
print(X_test_title_number_words_norm.shape, y_test.shape)
print("="*100)
```

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

## 1.5.10 encoding numerical features: essay_count

In [81]:

```
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['essay_number_words'].values.reshape(1,-1))

X_train_essay_number_words_norm =
normalizer.transform(X_train['essay_number_words'].values.reshape(-1,1))
X_cv_essay_number_words_norm = normalizer.transform(X_cv['essay_number_words'].values.reshape(-1,1)
)
X_test_essay_number_words_norm = normalizer.transform(X_test['essay_number_words'].values.reshape(
-1,1))



print("After vectorizations")
print(X_train_essay_number_words_norm.shape, y_train.shape)
print(X_cv_essay_number_words_norm.shape, y_cv.shape)
print(X_test_essay_number_words_norm.shape, y_test.shape)
print("="*100)
```

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

## 1.5.11 Concatinating all the features- tfidf

In [82]:

```
from scipy.sparse import hstack
X_tr = hstack((X_train_project_title_Tfidf,X_train_essay_Tfidf,
X_train_project_subject_categories_ohe, X_train_project_subject_subcategories_ohe,
X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_price_norm,
X_train_quantity_norm, X_train_num_prev_projects_norm, X_train_title_number_words_norm ,
X_train_essay_number_words_norm)).tocsr()
X_te = hstack((X_test_project_title_Tfidf,X_test_essay_Tfidf,
X_test_project_subject_categories_ohe, X_test_project_subject_subcategories_ohe, X_test_state_ohe,
X_test_teacher_ohe, X_test_grade_ohe, X_test_price_norm, X_test_quantity_norm,
X_test_num_prev_projects_norm, X_test_title_number_words_norm , X_test_essay_number_words_norm)).t
ocsr()
X_cr = hstack((X_cv_project_title_Tfidf,X_cv_essay_Tfidf, X_cv_project_subject_categories_ohe,
X_cv_project_subject_subcategories_ohe, X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe,
X_cv_price_norm, X_cv_quantity_norm, X_cv_num_prev_projects_norm, X_cv_title_number_words_norm , X
_cv_essay_number_words_norm)).tocsr()

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, 7615) (49041,)
(24155, 7615) (24155,)
(36052, 7615) (36052,)
====================================================================================================
```

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

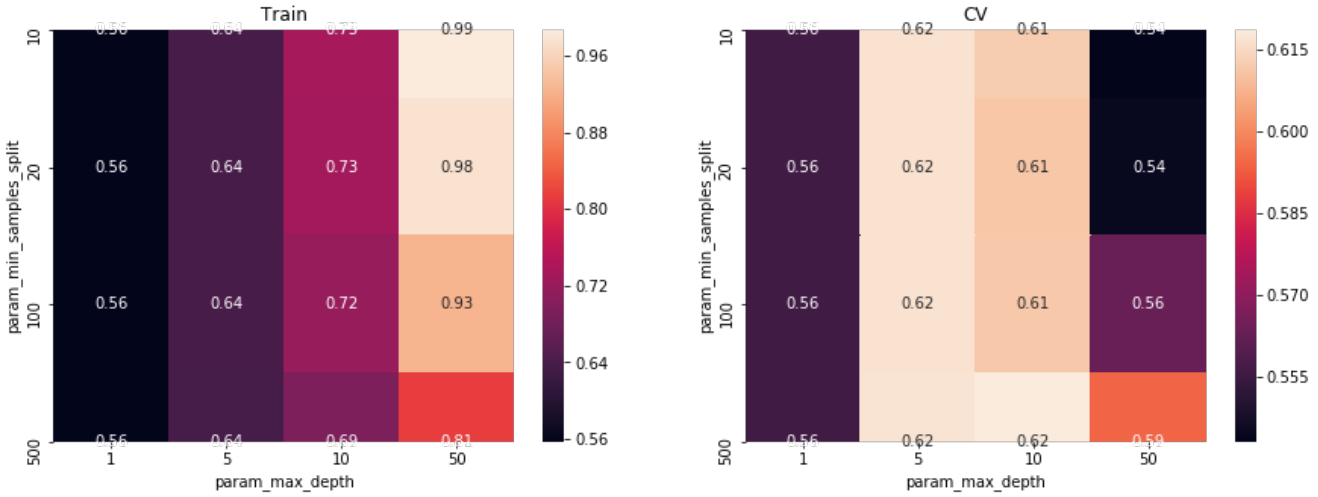
Apply Decision Tree on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

In [83]:

```
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
dectree = DecisionTreeClassifier(class_weight = 'balanced')
param = {'max_depth': [1, 5,10, 50], 'min_samples_split': [ 10, 20, 100, 500]}
clf = GridSearchCV(dectree, param, cv=3, scoring='roc_auc',return_train_score=True)
final = clf.fit(X_tr, y_train)
```

In [84]:

```
import seaborn as sns

maximum_score = pd.DataFrame(clf.cv_results_).groupby(['param_min_samples_split',
'param_max_depth']).max().unstack()[['mean_test_score','mean_train_score']]

fig, ax = plt.subplots(1,2, figsize=(15,5))
sns.heatmap(maximum_score.mean_train_score, annot = True, ax=ax[0])
sns.heatmap(maximum_score.mean_test_score, annot = True, ax=ax[1])
ax[0].set_title('Train')
ax[1].set_title('CV')
plt.show()
```




# Best Estimator and Best tune parameters

In [85]:

```
print(clf.best_estimator_)

print(clf.score(X_tr,y_train))
print(clf.score(X_te,y_test))
```

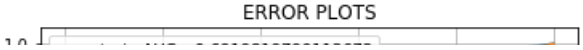
```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=10,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
0.6920293461225613
0.6187943486527919
```

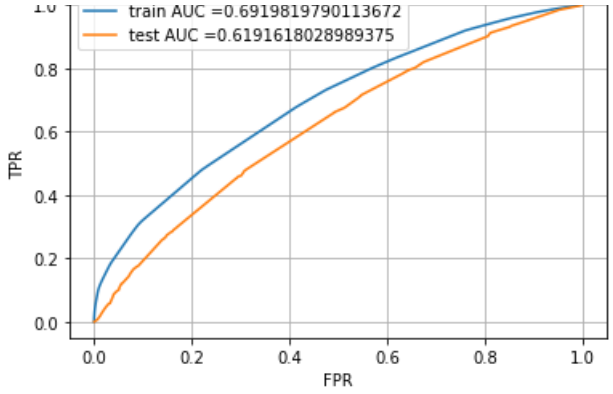
In [86]:

```
best_parameter=[{'max_depth':[10], 'min_samples_split':[500] }]
```

In [87]:

```
from sklearn.metrics import roc_curve, auc
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, auc
clf1= GridSearchCV( DecisionTreeClassifier(class_weight = 'balanced'),best_parameter)
clf2=DecisionTreeClassifier (class_weight = 'balanced',max_depth=10,min_samples_split=500)
clf1.fit(X_tr, y_train)

clf2.fit(X_tr, y_train)

yp_train = clf1.predict_proba(X_tr) [:,1]
yp_test = clf1.predict_proba(X_te) [:,1]

tr_fpr, tr_tpr, tr_thresh = roc_curve(y_train, yp_train)
te_fpr, te_tpr, te_thresh = roc_curve(y_test, yp_test)

plt.plot(tr_fpr, tr_tpr, label="train AUC ="+str(auc(tr_fpr, tr_tpr)))
plt.plot(te_fpr, te_tpr, label="test AUC ="+str(auc(te_fpr, te_tpr)))
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ERROR PLOTS")
plt.legend()
plt.grid()
plt.show()
```






# Confusion matrix

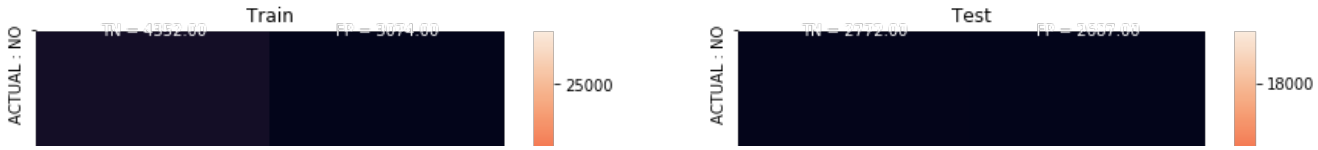
In [88]:

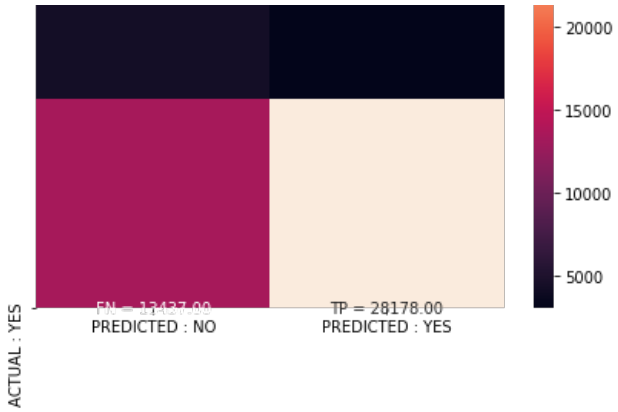
```
def predict(proba, thresh, fpr, tpr):
    t = thresh[np.argmax(fpr*(1-tpr))]
    print(" max value of tpr*(1-fpr)", np.round(max(tpr*(1-fpr)),2) , "for threshold", np.round(t,2
))
    pred = []
    global predictions1
    for i in proba:
        if i>=t:
            pred.append(1)
        else:
            pred.append(0)
    predictions1= pred
    return pred
```

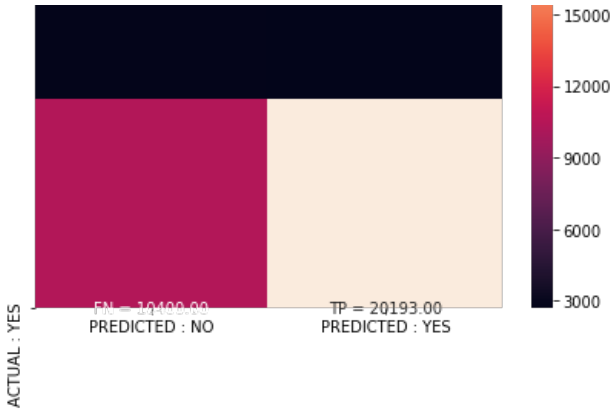
In [89]:

```
import seaborn as sns

con_train = confusion_matrix(y_train, predict(yp_train, tr_thresh, tr_fpr, tr_tpr))

con_test = confusion_matrix(y_test, predict(yp_test, te_thresh, te_fpr, te_tpr))

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

fig, ax = plt.subplots(1,2, figsize=(15,5))

train_label = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(),
con_train.flatten())])).reshape(2,2)

test_label = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(),c
on_test.flatten())])).reshape(2,2)

sns.heatmap(con_train,  xticklabels=['PREDICTED : NO', 'PREDICTED : YES'],yticklabels=['ACTUAL : N
O', 'ACTUAL : YES'], annot = train_label, fmt = '', ax=ax[0])

sns.heatmap(con_test, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'],yticklabels=['ACTUAL : NO'
, 'ACTUAL : YES'], annot = test_label, fmt = '', ax=ax[1])

ax[0].set_title('Train')

ax[1].set_title('Test')

plt.show()
```

```
 max value of tpr*(1-fpr) 0.4 for threshold 0.51
 max value of tpr*(1-fpr) 0.34 for threshold 0.51
```






## Visualizing Decision Tree

In [90]:

```
f1= vectorizer_cat.get_feature_names()

f2= vectorizer_subcat.get_feature_names()

f3= vectorizer_state.get_feature_names()

f4= vectorizer_grade.get_feature_names()

f5= vectorizer_teacher.get_feature_names()

f6= vectorizer_title_Tfidf.get_feature_names()

f7= vectorizer_essay_Tfidf.get_feature_names()
```

In [91]:

```
feat_tfidf = f1 + f2 + f3 + f4 + f5 + f6+ f7
```

In [92]:

```
feat_tfidf.append('price')
feat_tfidf.append('quantity')
feat_tfidf.append('teacher_number_of_previously_posted_projects')
```

## Analysis on the False positives

In [93]:

```
X_test['essay'].values[1]
```

Out[93]:

```
"My students come from low income homes.  We are a title one school with very little funding. My s
tudents are very sweet and want the opportunity to learn. My students need these supplies to be ab
le to do well in class. These supplies are essential for the students to succeed. They need the ch
ance to have the same education as everyone else.  I would like to try and provide that for them.
That is why my students need these supplies. \\r\\n\\r\\n  \\r\\n\\r\\nI teach at a Title 1 very l
ow economic school.  There are not a lot of funds for supplies, so these supplies would be used to
their fullest.  These supplies would give my students an opportunity to be able to do more hands o
n activity's. The students are so excited to learn and find out what fun stuff I have planned for
them for the day.  These things would be able to let me create even more things for them to do.  W
e try to get away from pencile and paper to do the majority of our learning.  Having these
supplies would allow us to be bale to have more fun with mathnannan"
```

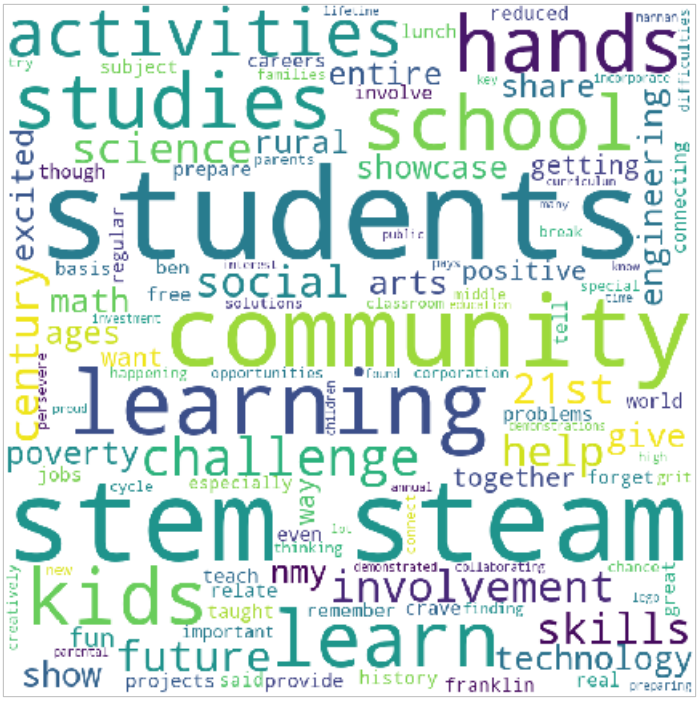
In [94]:

```
fp_list = []
```

```
for i in range(len(y_test)) :
    if (y_test.values[i] == 0) & (predictions1[i] == 1) :
        fp_list.append(i)
fp_essay = []
for i in fp_list :
    fp_essay.append(X_test['essay'].values[i])
```

In [95]:

```
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay :
    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 [96]:

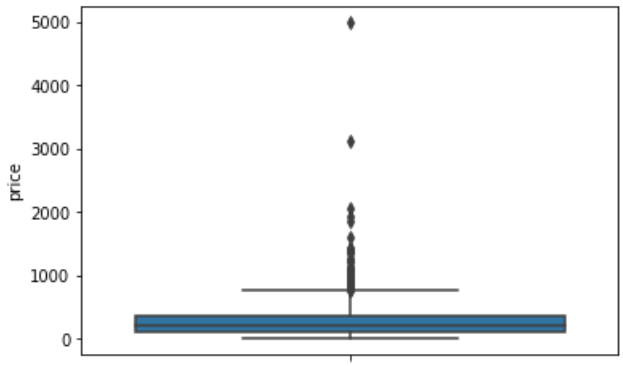
```
cols = X_test.columns
X_test_falsepos = pd.DataFrame(columns=cols)

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

In [97]:

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

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x127a0bf1d88>
```

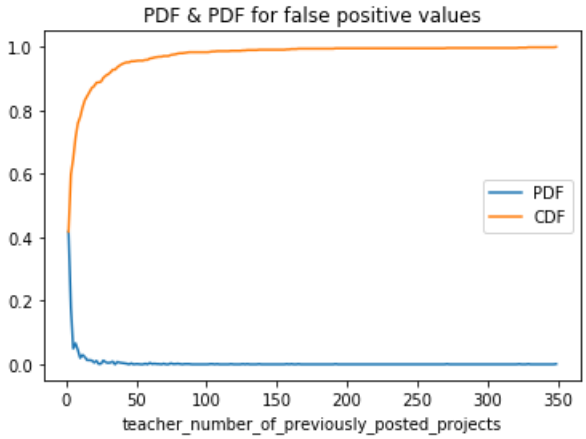



In [98]:

```
import matplotlib.pyplot as plt
```

In [99]:

```
plt.figure()
counts, bin_edges = np.histogram(X_test_falsepos['teacher_number_of_previously_posted_projects'],b
ins='auto', density=True)
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.title('PDF & PDF for false positive values')
plt.show()
```




# Concatinating all the features- w2v tfidf

In [100]:

```
from scipy.sparse import hstack
X_tr_w2v = hstack((tfidf_w2v_project_title_vectors_train,tfidf_w2v_vectors_train,
X_train_project_subject_categories_ohe, X_train_project_subject_subcategories_ohe,
X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe, X_train_price_norm,
X_train_quantity_norm, X_train_num_prev_projects_norm, X_train_title_number_words_norm ,
X_train_essay_number_words_norm)).tocsr()
X_te_w2v = hstack((tfidf_w2v_project_title_vectors_test,tfidf_w2v_vectors_test,
X_test_project_subject_categories_ohe, X_test_project_subject_subcategories_ohe, X_test_state_ohe,
X_test_teacher_ohe, X_test_grade_ohe, X_test_price_norm, X_test_quantity_norm,
X_test_num_prev_projects_norm, X_test_title_number_words_norm , X_test_essay_number_words_norm)).t
ocsr()
X_cr_w2v = hstack((tfidf_w2v_project_title_vectors_cv,tfidf_w2v_vectors_cv,
X_cv_project_subject_categories_ohe, X_cv_project_subject_subcategories_ohe, X_cv_state_ohe,
X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_price_norm, X_cv_quantity_norm, X_cv_num_prev_projects_norm
, X_cv_title_number_words_norm , X_cv_essay_number_words_norm)).tocsr()

print("Final Data matrix")
print(X_tr_w2v.shape, y_train.shape)
```

```
print(X_cr_w2v.shape, y_cv.shape)
print(X_te_w2v.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(49041, 1103) (49041,)
(24155, 1103) (24155,)
(36052, 1103) (36052,)
====================================================================================================
```

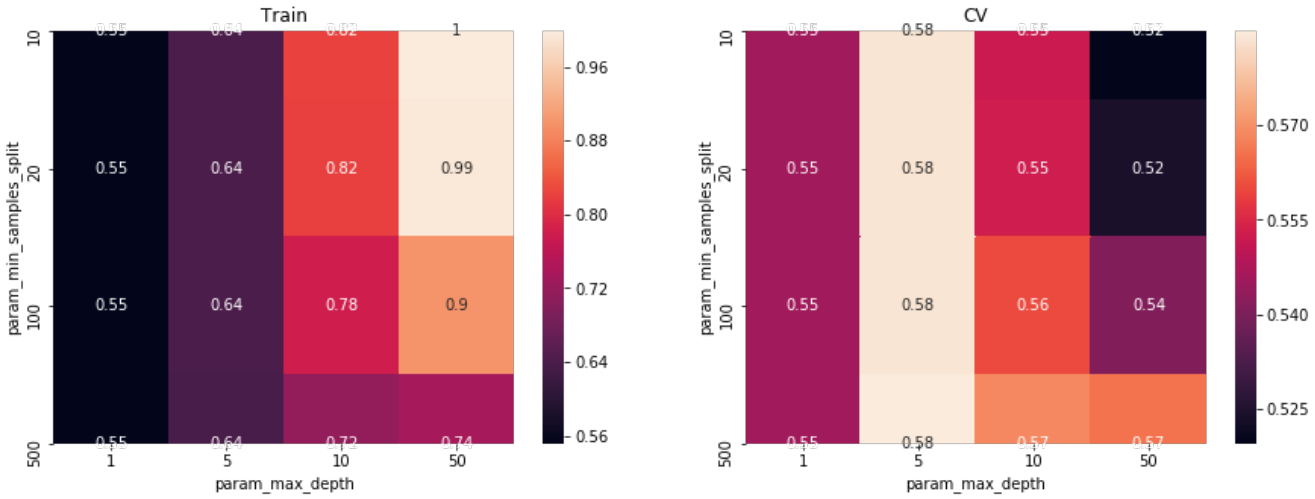
# Appling Decision Tree on different kind of featurization as mentioned in the instructions

In [101]:

```
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
dectree = DecisionTreeClassifier(class_weight = 'balanced')
param = {'max_depth': [1, 5,10, 50], 'min_samples_split': [ 10, 20, 100, 500]}
clf_w2v = GridSearchCV(dectree, param, cv=3, scoring='roc_auc',return_train_score=True)
final = clf_w2v.fit(X_tr_w2v, y_train)
```

In [102]:

```
import seaborn as sns

maximum_score = pd.DataFrame(clf_w2v.cv_results_).groupby(['param_min_samples_split',
'param_max_depth']).max().unstack()[['mean_test_score','mean_train_score']]

fig, ax = plt.subplots(1,2, figsize=(15,5))
sns.heatmap(maximum_score.mean_train_score, annot = True, ax=ax[0])
sns.heatmap(maximum_score.mean_test_score, annot = True, ax=ax[1])
ax[0].set_title('Train')
ax[1].set_title('CV')
plt.show()
```




# Best Estimator and Best tune parameters

In [103]:

```
print(clf_w2v.best_estimator_)

print(clf_w2v.score(X_tr_w2v,y_train))
print(clf_w2v.score(X_te_w2v,y_test))
```

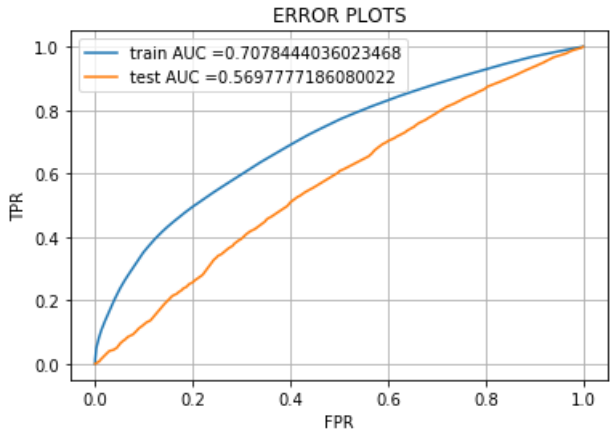
```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=5,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
0.6258421309647232
0.5871820893552323
```

In [104]:

```
best_parameter=[{'max_depth':[10], 'min_samples_split':[500] }]
```

In [105]:

```
from sklearn.metrics import roc_curve, auc
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, auc
clf1= GridSearchCV( DecisionTreeClassifier(class_weight = 'balanced'),best_parameter)
clf2=DecisionTreeClassifier (class_weight = 'balanced',max_depth=10,min_samples_split=500)
clf1.fit(X_tr_w2v, y_train)

clf2.fit(X_tr_w2v, y_train)

yp_train = clf1.predict_proba(X_tr_w2v) [:,1]
yp_test = clf1.predict_proba(X_te_w2v) [:,1]

tr_fpr, tr_tpr, tr_thresh = roc_curve(y_train, yp_train)
te_fpr, te_tpr, te_thresh = roc_curve(y_test, yp_test)

plt.plot(tr_fpr, tr_tpr, label="train AUC ="+str(auc(tr_fpr, tr_tpr)))
plt.plot(te_fpr, te_tpr, label="test AUC ="+str(auc(te_fpr, te_tpr)))
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ERROR PLOTS")
plt.legend()
plt.grid()
plt.show()
```




# Confusion matrix

In [106]:

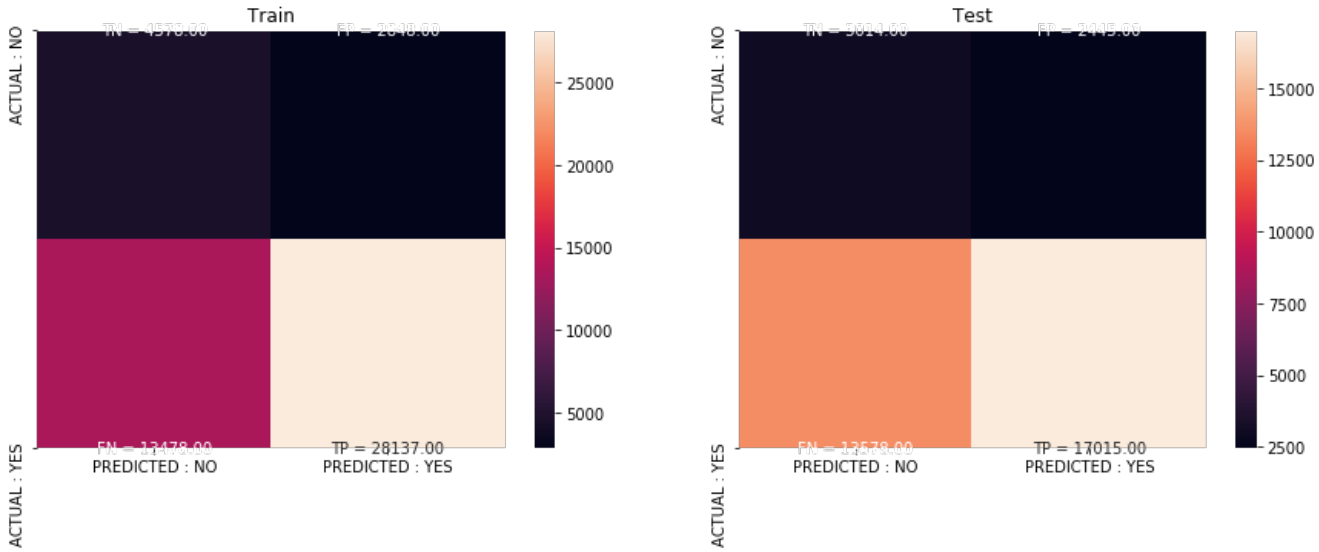
```
def predict(proba, thresh, fpr, tpr):
    t = thresh[np.argmax(fpr*(1-tpr))]
    print(" max value of tpr*(1-fpr)", np.round(max(tpr*(1-fpr)),2) , "for threshold", np.round(t,2
))
    pred = []
    global predictions1
    for i in proba:
        if i>=t:
            pred.append(1)
```

```
        else:
            pred.append(0)
    predictions1= pred
    return pred
```

In [107]:

```
import seaborn as sns

con_train = confusion_matrix(y_train, predict(yp_train, tr_thresh, tr_fpr, tr_tpr))

con_test = confusion_matrix(y_test, predict(yp_test, te_thresh, te_fpr, te_tpr))

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

fig, ax = plt.subplots(1,2, figsize=(15,5))

train_label = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(),
con_train.flatten())])).reshape(2,2)

test_label = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(),c
on_test.flatten())])).reshape(2,2)

sns.heatmap(con_train, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'],yticklabels=['ACTUAL : NO
', 'ACTUAL : YES'], annot = train_label, fmt = '', ax=ax[0])

sns.heatmap(con_test, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'],yticklabels=['ACTUAL : NO'
, 'ACTUAL : YES'], annot = test_label, fmt = '', ax=ax[1])

ax[0].set_title('Train')

ax[1].set_title('Test')

plt.show()
```

```
max value of tpr*(1-fpr) 0.42 for threshold 0.47
max value of tpr*(1-fpr) 0.31 for threshold 0.5
```




# Analysis on the False positives

In [108]:

```
X_test['essay'].values[1]
```

Out[108]:

```
"My students come from low income homes.  We are a title one school with very little funding. My s
tudents are very sweet and want the opportunity to learn. My students need these supplies to be ab
le to do well in class. These supplies are essential for the students to succeed. They need the ch
ance to have the same education as everyone else.  I would like to try and provide that for them.
That is why my students need these supplies. \\r\\n\\r\\n  \\r\\n\\r\\nI teach at a Title 1 very l
```

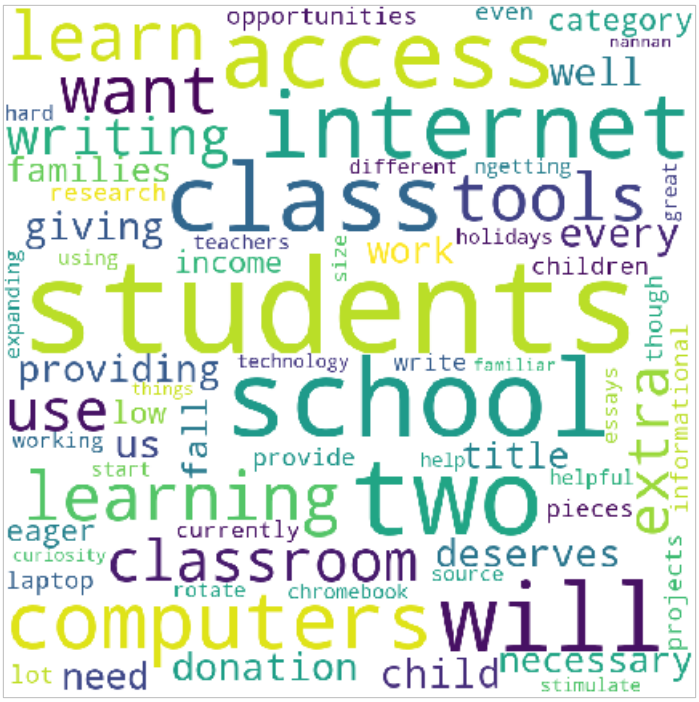
That is why my students need these supplies. \\r\\n\\r\\n  \\r\\n\\r\\nI teach at a little r very l
ow economic school.  There are not a lot of funds for supplies, so these supplies would be used to
their fullest.  These supplies would give my students an opportunity to be able to do more hands o
n activity's. The students are so excited to learn and find out what fun stuff I have planned for
them for the day.  These things would be able to let me create even more things for them to do.  W
e try to get away from pencile and paper to do the majority of our learning.  Having these
supplies would allow us to be bale to have more fun with mathnannan"

In [109]:

```
fp_list = []
for i in range(len(y_test)) :
    if (y_test.values[i] == 0) & (predictions1[i] == 1) :
        fp_list.append(i)
fp_essay = []
for i in fp_list :
    fp_essay.append(X_test['essay'].values[i])
```

In [110]:

```
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay :
    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 [111]:

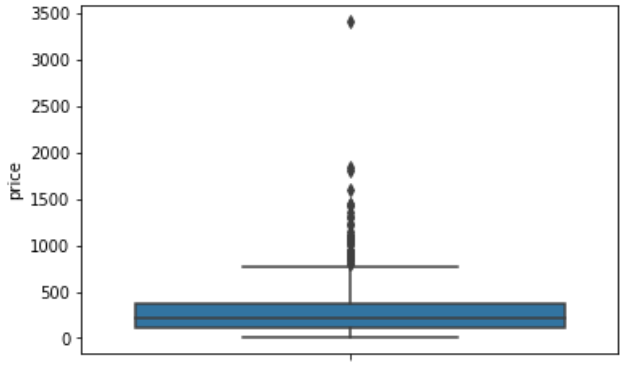
```
cols = X_test.columns
X_test_falsepos = pd.DataFrame(columns=cols)

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

In [112]:

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

Out[112]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x127a2340848>
```

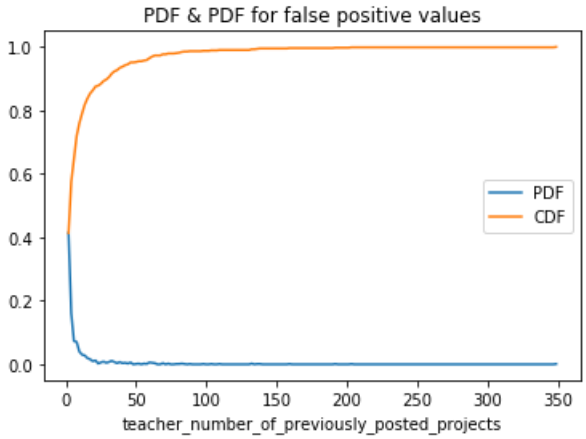



In [113]:

```
import matplotlib.pyplot as plt
```

In [114]:

```
plt.figure()
counts, bin_edges = np.histogram(X_test_falsepos['teacher_number_of_previously_posted_projects'],b
ins='auto', density=True)
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.title('PDF & PDF for false positive values')
plt.show()
```




## Getting top features using feature_importances_

In [115]:

```
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]]
```

```
    return X[:,model.best_estimator_.feature_importances_.argsort()[::-1][:k]]
```

In [116]:

```
xtrain = X_tr.tocsr()
```

In [117]:

```
X_tr = selectKImportance(clf, xtrain,5000)
X_te = selectKImportance(clf, X_te, 5000)

print(X_tr.shape)
print(X_te.shape)
```

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

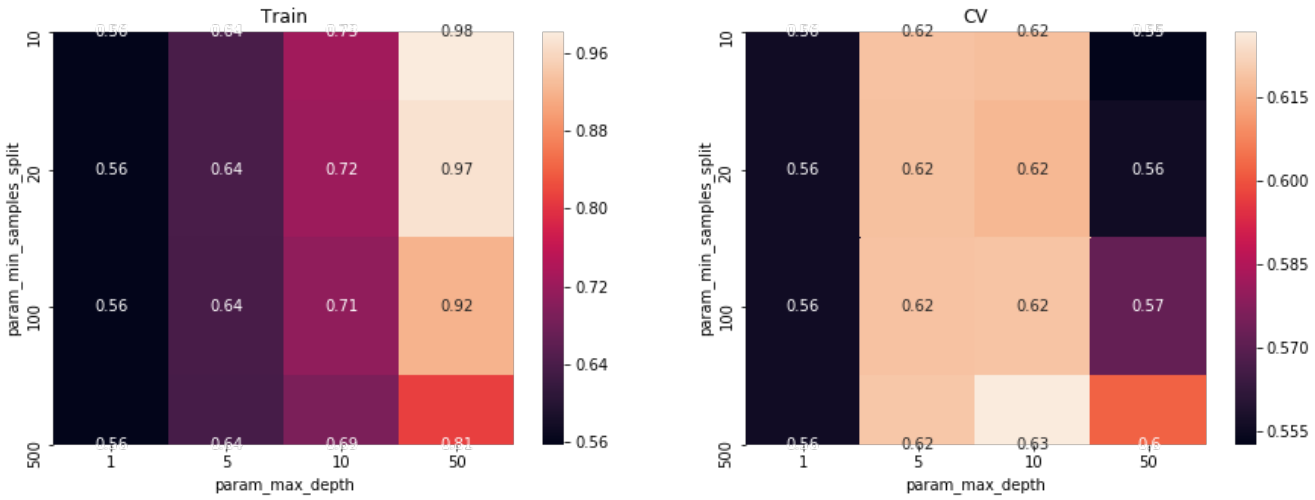
## Appling Decision Tree on different kind of featurization as mentioned in the instructions

In [118]:

```
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
dectree = DecisionTreeClassifier(class_weight = 'balanced')
param = {'max_depth': [1, 5,10, 50], 'min_samples_split': [ 10, 20, 100, 500]}
clf = GridSearchCV(dectree, param, cv=3, scoring='roc_auc',return_train_score=True)
final = clf.fit(X_tr, y_train)
```

In [119]:

```
import seaborn as sns

maximum_score = pd.DataFrame(clf.cv_results_).groupby(['param_min_samples_split',
'param_max_depth']).max().unstack()[['mean_test_score','mean_train_score']]

fig, ax = plt.subplots(1,2, figsize=(15,5))
sns.heatmap(maximum_score.mean_train_score, annot = True, ax=ax[0])
sns.heatmap(maximum_score.mean_test_score, annot = True, ax=ax[1])
ax[0].set_title('Train')
ax[1].set_title('CV')
plt.show()
```




## Best Estimator and Best tune parameters

## Best Estimator and Best tune parameters

In [120]:

```
print(clf.best_estimator_)

print(clf.score(X_tr,y_train))
print(clf.score(X_te,y_test))
```

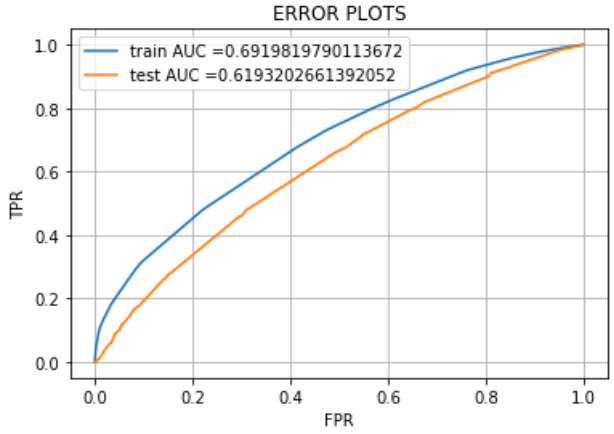
```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_depth=10,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=500,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
0.6919819790113672
0.6188440411250087
```

In [121]:

```
best_parameter=[{'max_depth':[10], 'min_samples_split':[500] }]
```

In [122]:

```
from sklearn.metrics import roc_curve, auc
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import roc_curve, auc
clf1= GridSearchCV( DecisionTreeClassifier(class_weight = 'balanced'),best_parameter)
clf2=DecisionTreeClassifier (class_weight = 'balanced',max_depth=10,min_samples_split=500)
clf1.fit(X_tr, y_train)

clf2.fit(X_tr, y_train)

yp_train = clf1.predict_proba(X_tr) [:,1]
yp_test = clf1.predict_proba(X_te) [:,1]

tr_fpr, tr_tpr, tr_thresh = roc_curve(y_train, yp_train)
te_fpr, te_tpr, te_thresh = roc_curve(y_test, yp_test)

plt.plot(tr_fpr, tr_tpr, label="train AUC ="+str(auc(tr_fpr, tr_tpr)))
plt.plot(te_fpr, te_tpr, label="test AUC ="+str(auc(te_fpr, te_tpr)))
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ERROR PLOTS")
plt.legend()
plt.grid()
plt.show()
```




## Confusion matrix

In [123]:

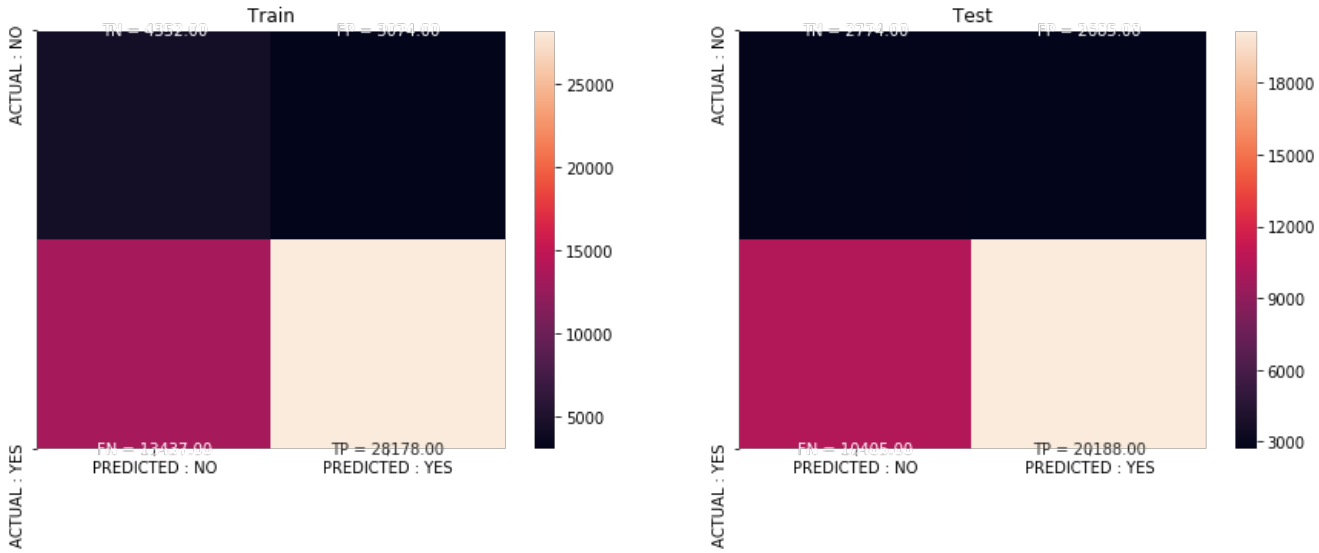
```
def predict(proba, thresh, fpr, tpr):
    t = thresh[np.argmax(fpr*(1-tpr))]
```

```
    print(" max value of tpr*(1-fpr)", np.round(max(tpr*(1-fpr)),2) , "for threshold", np.round(t,2
))
    pred = []
    global predictions1
    for i in proba:
        if i>=t:
            pred.append(1)
        else:
            pred.append(0)
    predictions1= pred
    return pred
```

In [124]:

```
import seaborn as sns

con_train = confusion_matrix(y_train, predict(yp_train, tr_thresh, tr_fpr, tr_tpr))

con_test = confusion_matrix(y_test, predict(yp_test, te_thresh, te_fpr, te_tpr))

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))

fig, ax = plt.subplots(1,2, figsize=(15,5))

train_label = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(),
con_train.flatten())])).reshape(2,2)

test_label = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key.flatten(),c
on_test.flatten())])).reshape(2,2)

sns.heatmap(con_train, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'],yticklabels=['ACTUAL : NO
', 'ACTUAL : YES'], annot = train_label, fmt = '', ax=ax[0])

sns.heatmap(con_test, xticklabels=['PREDICTED : NO', 'PREDICTED : YES'],yticklabels=['ACTUAL : NO'
, 'ACTUAL : YES'], annot = test_label, fmt = '', ax=ax[1])

ax[0].set_title('Train')

ax[1].set_title('Test')

plt.show()
```

```
 max value of tpr*(1-fpr) 0.4 for threshold 0.51
 max value of tpr*(1-fpr) 0.34 for threshold 0.51
```




# Analysis on the False positives

In [125]:

```
X_test['essay'].values[1]
```

Out[125]:

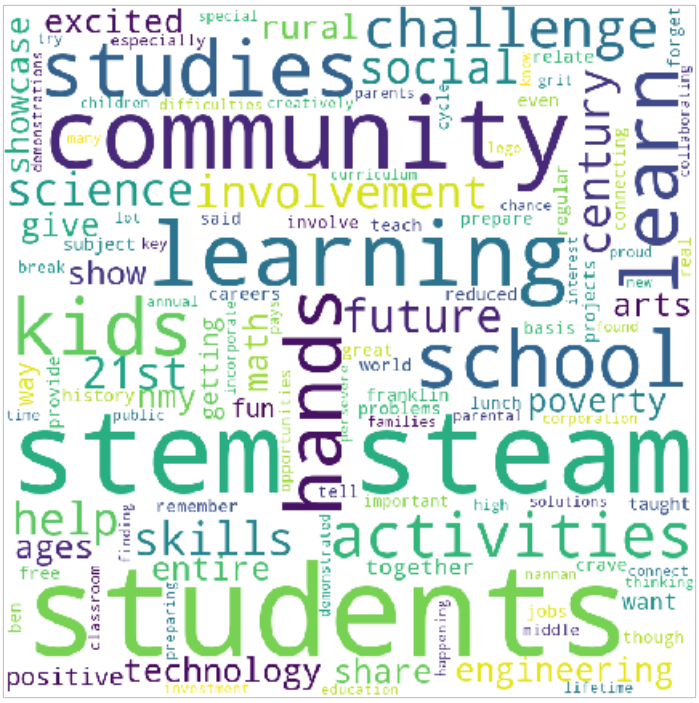
"My students come from low income homes.  We are a title one school with very little funding. My s
tudents are very sweet and want the opportunity to learn. My students need these supplies to be ab
le to do well in class. These supplies are essential for the students to succeed. They need the ch
ance to have the same education as everyone else.  I would like to try and provide that for them.
That is why my students need these supplies. \\r\\n\\r\\n  \\r\\n\\r\\nI teach at a Title 1 very l
ow economic school.  There are not a lot of funds for supplies, so these supplies would be used to
their fullest.  These supplies would give my students an opportunity to be able to do more hands o
n activity's. The students are so excited to learn and find out what fun stuff I have planned for
them for the day.  These things would be able to let me create even more things for them to do.  W
e try to get away from pencile and paper to do the majority of our learning.  Having these
supplies would allow us to be bale to have more fun with mathnannan"

In [126]:

```
fp_list = []
for i in range(len(y_test)) :
    if (y_test.values[i] == 0) & (predictions1[i] == 1) :
        fp_list.append(i)
fp_essay = []
for i in fp_list :
    fp_essay.append(X_test['essay'].values[i])
```

In [127]:

```
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay :
    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 [128]:

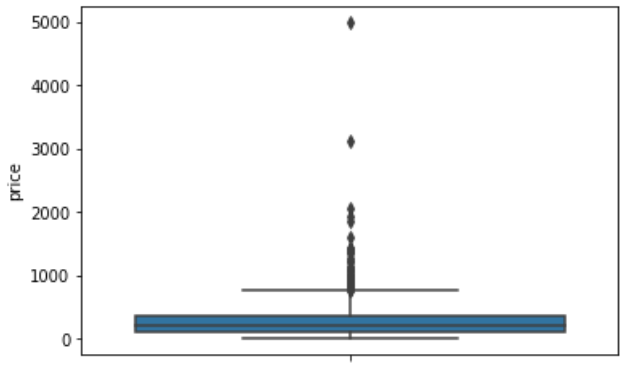
```
cols = X_test.columns
X_test_falsepos = pd.DataFrame(columns=cols)

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

In [129]:

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

Out[129]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x127200d1a08>
```

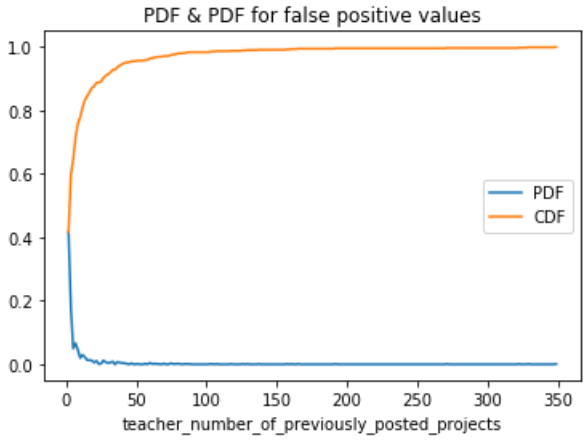



In [130]:

```
import matplotlib.pyplot as plt
```

In [131]:

```
plt.figure()
counts, bin_edges = np.histogram(X_test_falsepos['teacher_number_of_previously_posted_projects'],b
ins='auto', density=True)
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.title('PDF & PDF for false positive values')
plt.show()
```




# Summary

In [132]:

```
from prettytable import PrettyTable
tb = PrettyTable()
tb.field_names= (" Vectorizer ", " Max_depth ", " Min_sample_split "," Test -AUC ")
tb.add_row([" TfIdf", 10 , 500 ,61.91])
tb.add_row(["TfIdf_w2v", 10 , 500 ,56.97])
tb.add_row(["Top Features(Tfidf)", 10, 500 ,61.93 ])
print(tb.get_string(titles = "Observations"))
```

```
+---------------------+-------------+--------------------+-------------+
|      Vectorizer     |  Max_depth  |  Min_sample_split  |  Test -AUC  |
+---------------------+-------------+--------------------+-------------+
|         TfIdf       |      10     |        500         |    61.91    |
|      TfIdf_w2v      |      10     |        500         |    56.97    |
| Top Features(Tfidf) |      10     |        500         |    61.93    |
+---------------------+-------------+--------------------+-------------+
```

In [ ]:

```
```