Document Created Date:16-05-2019

Understanding the Data and DataSource - DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	
`project_id`	A unique identifier for the proposed project. **Example**
`project_title`	Title of the projec • `Art Will Make Y` • `First`

Feature	
project_grade_category	Grade level of students for which the project is targeted. One of t enumera • Grad • G • G • Gr
project_subject_categories	One or more (comma-separated) subject categories for the pro following enumerated li • Applied • Care • Health • History • Literacy & • Math • Music & • Spec •
school_state	• Music & • Literacy & Language, Math State where school is located (Two-letter U.S. (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Pos **Ex**
project_subject_subcategories	One or more (comma-separated) subject subcategories fo • • Literature & Writing, Social
project_resource_summary	An explanation of the resources needed for the projec • My students need hands on literacy materials t sensory nee
project_essay_1	First applic
project_essay_2	Second applic
project_essay_3	Third applic
project_essay_4	Fourth applic
project_submitted_datetime	Datetime when project application was submitted. **Example:** 2(12:
teacher_id	A unique identifier for the teacher of the proposed proje bdf8baa8fedef6bfeec7ae4
teacher_prefix	Teacher's title. One of the following enumera • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the sa **I**

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

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

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

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

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

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

Notes on the Essay Data

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

- **project_essay_1:** "Introduce us to your classroom"
- **project_essay_2:** "Tell us more about your students"
- **project_essay_3:** "Describe how your students will use the materials you're requesting"
- **project_essay_3:** "Close by sharing why your project will make a difference"

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

- **project_essay_1:** "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- **project_essay_2:** "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

Decision Tree

Step by Step Procedure

- Understanding the Businessreal world problem
- Loading the data
- Preprocessing the data(based on the type of data = categorical , text, Numarical)
- Preprocessing data includes (removing outliers, impute missung values, cleaning data, remove spacial character, etc..)
- Split the data into train, cv, test(random splitting)

- Vectorization data (one hot encoding)
- Vectorizing text data(bow, tfidf, avgw2v, tfidf weighted w2v)
- Vectorizing numarical - Normalizer
- Applying Desition Trees Model on top of the features
- Contactinating all the type of features(cat + num + selected text features)
- Hyperparameter tuning to find th best estimator(GridSearchCV) and Ploting heatmaps
- Train the Desition Trees Model using best hyperparameter and ploting auc roc-curve
- Ploting confusion matrix(heatmaps)
- Graphviz visualization of Decision Tree
- Finding the False Possitive points
- Ploting wordcloud with the words of essay text of these false positive data points
- Ploting Box plot with price of false possitive points
- PDF & CDF with teacher_number_of_previously_posted_projects false possitive points
- Getting top 5k features using feature_importances_with TFIDF
- Hyperparameter tuning to find th best estimator(GridSearchCV) and Ploting heatmaps
- Train the Desition Trees Model using best hyperparameter and ploting auc roc-curve
- Ploting confusion matrix(heatmaps)
- Graphviz visualization of Decision Tree
- Finding the False Possitive points
- Ploting wordcloud with the words of essay text of these false positive data points
- Ploting Box plot with price of false possitive points
- PDF & CDF with teacher_number_of_previously_posted_projects false possitive points
- Observation on overall model performences(Conclusion)
- Ploting the performences by table format.

```
C:\Users\Ramesh Battu> import required libraries
```

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os
```

1.1 Reading Data

In [2]:

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

In [3]:

```
print("Number of data points in train data", project_data.shape)
print('-'*88)
print("The attributes of data :", project_data.columns.values)
print('-'*88)
```

```
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]:

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

#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/40840
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)

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

project_data.head(2)
```

Out[4]:

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

In [5]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
print('-'*60)
resource_data.head(2)
```

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

Out[5]:

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

1.1.1 Preprocessing of project_subject_categories

In [6]:

```
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
catogories = list(project_data['project_subject_categories'].values)

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

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

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

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

1.1.2 Preprocessing of project_subject_subcategories

In [7]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth

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

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

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

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

1.1.3 Preprocessing of `school_state`

In [8]:

```
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
school_state_catogories = list(project_data['school_state'].values)
cat_list = []
for i in school_state_catogories:
    temp = "" # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
project_data['school_state'] = cat_list

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

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

1.1.4 Preprocessing of `teacher_prefix`

In [9]:

```
# citation code :https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['teacher_prefix'].value_counts().index[(
teacher_prefix_catogories = list(project_data['teacher_prefix'].values)
# Citation code : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit
# To convert the data type object to unicode string : used """astype('U')""" code from t
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
cat_list = []
for i in teacher_prefix_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['teacher_prefix'] = cat_list

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

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
teacher_prefix_dict = dict(my_counter)
sorted_teacher_prefix_dict = dict(sorted(teacher_prefix_dict.items(), key=lambda kv: kv
```

1.1.5 Preprocessing of `project_grade_category`

In [10]:

```
project_grade_catogories = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
cat_list = []
for i in project_grade_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['project_grade_category'] = cat_list

#link : https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['project_grade_category'].value_counts(

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

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

1.2. Text Preprocessing

1.2.1 Text Preprocessing of essay

In [11]:

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

In [12]:

```
project_data.head(1)
```

Out[12]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
55660	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	00

In [13]:

```
# printing some random reviews
print(project_data['essay'].values[1])
print("="*125)
print(project_data['essay'].values[125])
print("="*125)
print(project_data['essay'].values[2020])
print("="*125)
print(project_data['essay'].values[40020])
print("="*125)
print(project_data['essay'].values[99999])
print("="*125)
```

```
Imagine being 8-9 years old. You're in your third grade classroom. You see
bright lights, the kid next to you is chewing gum, the birds are making no
ise, the street outside is buzzing with cars, it's hot, and your teacher i
s asking you to focus on learning. Ack! You need a break! So do my student
s.Most of my students have autism, anxiety, another disability, or all of
the above. It is tough to focus in school due to sensory overload or emoti
ons. My students have a lot to deal with in school, but I think that makes
them the most incredible kids on the planet. They are kind, caring, and sy
mpathetic. They know what it's like to be overwhelmed, so they understand
when someone else is struggling. They are open-minded and compassionate. T
hey are the kids who will someday change the world.It is tough to do more
than one thing at a time. When sensory overload gets in the way, it is the
hardest thing in the world to focus on learning. My students need many bre
aks throughout the day, and one of the best items we've used is a Boogie B
oard. If we had a few in our own classroom, my students could take a break
exactly when they need one, regardless of which other rooms in the school
are occupied. Many of my students need to do something with their hands in
order to focus on the task at hand. Putty will give the sensory input they
need in order to focus, it will calm them when they are overloaded, it wil
l help improve motor skills, and it will make school more fun.When my stud
ents are able to calm themselves down, they are ready to learn. When they
are able to focus, they will learn more and retain more. They will get the
sensory input they need and it will prevent meltdowns (which are scary for
everyone in the room). This will lead to a better, happier classroom commu
nity that is able to learn the most they can in the best way possible.
==========================================================================
==========================================================
Seventh and eighth grade students at my school are getting to use the scho
ol's science lab for the first time this year. It is my hope that science
will quickly become their favorite subject when they realize it is not jus
t a subject, but everything in the world around them.My students are the f
uture leaders of their community. They are learning to set an example of e
xcellence and service in all they do. Scholars in our middle school progra
m are working hard to gain the skills they need to succeed in today's comp
etitive world, but are doing so in an environment that nurtures individual
s and encourages peace and thoughtfulness.Students will each have a binder
where they can keep assignments for their classes.  The Middle School Team
will help the students learn to keep their work for each class in a separa
te tab and incorporate a color-coded system. Kids will be able to personal
ize their binders and have them available at all times to prevent assignme
nts from getting lost.Adolescents are notoriously forgetful and disorganiz
ed. Our team hopes that creating a fail-safe organization plan, students w
ill be able to keep track of their work and important papers.
==========================================================================
==========================================================
I have long dreamed of teaching Angels in America, a play for my AP studen
ts that stimulated their thoughts and understand the universal promise of
```

the American dream.My students come from extremely diverse backgrounds.\r
\n\r\nThere are students who have fled war-torn countries such as the Ukra
ine, to first generation Mexican-Americans, to students dealing with Asber
gers and even a student who is in the advanced stages of Muscular Dystroph
y. Through all their struggles, they are extremely resilient and come to s
chool every day with hopes of a better future.In class we will be reading
\"Angels in America\" together and discussing in Socratic seminars and wri
ting papers on themes such as: visions of America, magical realism, the ne
ed for a sense of community in our lives, and how caustic and demeaning st
ereotyping can be.AP students are at an age in their lives where they are
ready to see the world through many lenses. These unique perspectives mak
e they not only better readers and writers, but also more prepared for the
world they are entering.
==========
==========
A typical day in my classroom is filled with active, fun-loving, energetic
2nd graders. They are eager to learn, ask questions, and explore. One of
my biggest jobs as their teacher is to keep them actively engaged in the l
earning experience. My students are 2nd graders who have the desire to gai
n as much knowledge as they can. They have a wide range of learning style
s and abilities, and all have the desire to learn and create with technolo
gy. Our school is in a very diverse, middle class, hard working neighborh
ood. Our families value education and expect a high level of rigor. \r\n
The students in my 2nd grade classroom need a MacBook Air laptop to suppor
t their daily learning. The laptop will provide students with opportuniti
es to take ownership and control of their own learning and also explore th
rough technology. Having a laptop computer in the classroom will also pro
vide them opportunities to engage in some of the lessons at their own pace
and effectively collaborate with other classmates. It will give them acce
ss to up to date information quickly and easily, read and store hundreds o
f iBooks, allow them to work on computer coding, and use applications and
software to publish completed books. The list of possibilities are endles
s!Mrs.Mrs.
==========
==========
My classroom consists of twenty-two amazing sixth graders from different c
ultures and backgrounds. They are a social bunch who enjoy working in part
ners and working with groups. They are hard-working and eager to head to m
iddle school next year. My job is to get them ready to make this transitio
n and make it as smooth as possible. In order to do this, my students need
to come to school every day and feel safe and ready to learn. Because they
are getting ready to head to middle school, I give them lots of choice- ch
oice on where to sit and work, the order to complete assignments, choice o
f projects, etc. Part of the students feeling safe is the ability for them
to come into a welcoming, encouraging environment. My room is colorful and
the atmosphere is casual. I want them to take ownership of the classroom b
ecause we ALL share it together. Because my time with them is limited, I w
ant to ensure they get the most of this time and enjoy it to the best of t
heir abilities.Currently, we have twenty-two desks of differing sizes, yet
the desks are similar to the ones the students will use in middle school.
We also have a kidney table with crates for seating. I allow my students t
o choose their own spots while they are working independently or in group
s. More often than not, most of them move out of their desks and onto the
crates. Believe it or not, this has proven to be more successful than maki
ng them stay at their desks! It is because of this that I am looking towar
d the “Flexible Seating” option for my classroom.\r\n The students look fo
rward to their work time so they can move around the room. I would like to
get rid of the constricting desks and move toward more “fun” seating optio
ns. I am requesting various seating so my students have more options to si
t. Currently, I have a stool and a papasan chair I inherited from the prev
ious sixth-grade teacher as well as five milk crate seats I made, but I wo

```
uld like to give them more options and reduce the competition for the “goo
d seats”. I am also requesting two rugs as not only more seating options b
ut to make the classroom more welcoming and appealing. In order for my stu
dents to be able to write and complete work without desks, I am requesting
a class set of clipboards. Finally, due to curriculum that requires groups
to work together, I am requesting tables that we can fold up when we are n
ot using them to leave more room for our flexible seating options.\r\nI kn
ow that with more seating options, they will be that much more excited abo
ut coming to school! Thank you for your support in making my classroom one
students will remember forever!Mrs.Mrs.
==========================================================================
==================================================
```

In [14]:

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

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

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

In [15]:

```
sent = decontracted(project_data['essay'].values[2020])
print(sent)
print("="*120)
```

```
I have long dreamed of teaching Angels in America, a play for my AP studen
ts that stimulated their thoughts and understand the universal promise of
the American dream.My students come from extremely diverse backgrounds.\r
\n\r\nThere are students who have fled war-torn countries such as the Ukra
ine, to first generation Mexican-Americans, to students dealing with Asber
gers and even a student who is in the advanced stages of Muscular Dystroph
y. Through all their struggles, they are extremely resilient and come to s
chool every day with hopes of a better future.In class we will be reading
\"Angels in America\" together and discussing in Socratic seminars and wri
ting papers on themes such as: visions of America, magical realism, the ne
ed for a sense of community in our lives, and how caustic and demeaning st
ereotyping can be.AP students are at an age in their lives where they are
ready to see the world through many lenses.  These unique perspectives mak
e they not only better readers and writers, but also more prepared for the
world they are entering.
==========================================================================
============================================
```

In [16]:

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

```
I have long dreamed of teaching Angels in America a play for my AP student
s that stimulated their thoughts and understand the universal promise of t
he American dream My students come from extremely diverse backgrounds r n
r nThere are students who have fled war torn countries such as the Ukraine
to first generation Mexican Americans to students dealing with Asbergers a
nd even a student who is in the advanced stages of Muscular Dystrophy Thro
ugh all their struggles they are extremely resilient and come to school ev
ery day with hopes of a better future In class we will be reading Angels i
n America together and discussing in Socratic seminars and writing papers
on themes such as visions of America magical realism the need for a sense
of community in our lives and how caustic and demeaning stereotyping can b
e AP students are at an age in their lives where they are ready to see the
world through many lenses These unique perspectives make they not only bet
ter readers and writers but also more prepared for the world they are ente
ring
```

In [17]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ["a","about","above","after","again","against","ain","all","am","an","and","
            "as","at","be","because","been","before","being","below","between","both",
            "d","did","didn","didn't","do","does","doesn","doesn't","doing","don","don
            "for","from","further","had","hadn","hadn't","has","hasn","hasn't","have",
            "here","hers","herself","him","himself","his","how","i","if","in","into","
            "itself","just","ll","m","ma","me","mightn","mightn't","more","most","must
            "needn't","no","nor","not","now","o","of","off","on","once","only","or","o
            "out","over","own","re","s","same","shan","shan't","she","she's","should",
            "so","some","such","t","than","that","that'll","the","their","theirs","the
            "these","they","this","those","through","to","too","under","until","up","v
            "we","were","weren","weren't","what","when","where","which","while","who",
            "won't","wouldn","wouldn't","y","you","you'd","you'll","you're","you've","
            "yourselves","could","he'd","he'll","he's","here's","how's","i'd","i'll","
            "she'd","she'll","that's","there's","they'd","they'll","they're","they've"
            "what's","when's","where's","who's","why's","would","able","abst","accorda
            "across","act","actually","added","adj","affected","affecting","affects","
            "along","already","also","although","always","among","amongst","announce",
            "anymore","anyone","anything","anyway","anyways","anywhere","apparently","
            "around","aside","ask","asking","auth","available","away","awfully","b","b
            "becoming","beforehand","begin","beginning","beginnings","begins","behind"
            "beyond","biol","brief","briefly","c","ca","came","cannot","can't","cause"
            "co","com","come","comes","contain","containing","contains","couldnt","dat
            "due","e","ed","edu","effect","eg","eight","eighty","either","else","elsewh
            "especially","et","etc","even","ever","every","everybody","everyone","ever
            "f","far","ff","fifth","first","five","fix","followed","following","follow
            "found","four","furthermore","g","gave","get","gets","getting","give","giv
            "gone","got","gotten","h","happens","hardly","hed","hence","hereafter","he
            "hes","hi","hid","hither","home","howbeit","however","hundred","id","ie","
            "importance","important","inc","indeed","index","information","instead","i
            "it'll","j","k","keep","keeps","kept","kg","km","know","known","knows","l"
            "later","latter","latterly","least","less","lest","let","lets","like","lik
            "'ll","look","looking","looks","ltd","made","mainly","make","makes","many"
            "meantime","meanwhile","merely","mg","might","million","miss","ml","moreov
            "mug","must","n","na","name","namely","nay","nd","near","nearly","necessar
            "neither","never","nevertheless","new","next","nine","ninety","nobody","no
            "normally","nos","noted","nothing","nowhere","obtain","obtained","obviousl
            "omitted","one","ones","onto","ord","others","otherwise","outside","overal
            "particular","particularly","past","per","perhaps","placed","please","plus
            "potentially","pp","predominantly","present","previously","primarily","pro
            "provides","put","q","que","quickly","quite","qv","r","ran","rather","rd",
            "recently","ref","refs","regarding","regardless","regards","related","rela
            "resulted","resulting","results","right","run","said","saw","say","saying"
            "seeing","seem","seemed","seeming","seems","seen","self","selves","sent","
            "shes","show","showed","shown","showns","shows","significant","significant
            "six","slightly","somebody","somehow","someone","somethan","something","so
            "somewhere","soon","sorry","specifically","specified","specify","specifyin
            "sub","substantially","successfully","sufficiently","suggest","sup","sure"
            "tends","th","thank","thanks","thanx","thats","that've","thence","thereaft
            "therein","there'll","thereof","therere","theres","thereto","thereupon","t
            "thou","though","thoughh","thousand","throug","throughout","thru","thus","
            "toward","towards","tried","tries","truly","try","trying","ts","twice","tw
            "unless","unlike","unlikely","unto","upon","ups","us","use","used","useful
            "using","usually","v","value","various","'ve","via","viz","vol","vols","vs
            "wed","welcome","went","werent","whatever","what'll","whats","whence","whe
            "whereby","wherein","wheres","whereupon","wherever","whether","whim","whit
            "who'll","whomever","whos","whose","widely","willing","wish","within","wit
            "wouldnt","www","x","yes","yet","youd","youre","z","zero","a's","ain't","a
```

```
            "appreciate","appropriate","associated","best","better","c'mon","c's","can
            "consequently","consider","considering","corresponding","course","currently
            "entirely","exactly","example","going","greetings","hello","help","hopeful
            "indicated","indicates","inner","insofar","it'd","keep","keeps","novel","pr
            "secondly","sensible","serious","seriously","sure","t's","third","thorough
            "wonder"]
```

In [18]:

```
%time
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = 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_essays.append(sent.lower().strip())
```

```
Wall time: 0 ns

100%|████████████████████████████████████████████████████████████████████████
████| 109248/109248 [07:10<00:00, 253.97it/s]
```

In [19]:

```
# after preprocesing
preprocessed_essays[40020]
```

Out[19]:

```
'typical day classroom filled active fun loving energetic 2nd graders eage
r learn questions explore biggest jobs teacher actively engaged learning e
xperience students 2nd graders desire gain knowledge wide range learning s
tyles abilities desire learn create technology school diverse middle class
hard working neighborhood families education expect high level rigor stude
nts 2nd grade classroom macbook air laptop support daily learning laptop p
rovide students opportunities ownership control learning explore technolog
y laptop computer classroom provide opportunities engage lessons pace effe
ctively collaborate classmates access easily read store hundreds ibooks wo
rk computer coding applications software publish completed books list poss
ibilities endless'
```

1.2.2 Text Preprocessing of `project_title`

In [20]:

```
print(project_data['project_title'].tail(1))
```

```
78306    News for Kids
Name: project_title, dtype: object
```

In [21]:

```
# printing some random title texts
print(project_data['project_title'].values[19])
print('--'*19)
print(project_data['project_title'].values[196])
print('--'*19)
print(project_data['project_title'].values[1969])
print('--'*19)
print(project_data['project_title'].values[99999])
print('--'*19)
```

```
Choice Novels for Freshman Students are Needed!!!!!!
------------------------------------
Pre-K STEM
------------------------------------
Divide and Color!
------------------------------------
Turning to Flexible Seating: One Sixth-Grade Class's Journey to Freedom
------------------------------------
```

In [22]:

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

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

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

In [23]:

```
sent = decontracted(project_data['project_title'].values[99999])
print(sent)
print("="*120)
```

```
Turning to Flexible Seating: One Sixth-Grade Class is Journey to Freedom
========================================================================
========================================
```

In [24]:

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

```
Turning to Flexible Seating: One Sixth-Grade Class is Journey to Freedom
```

In [25]:

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

```
Turning to Flexible Seating One Sixth Grade Class is Journey to Freedom
```

In [26]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ["a","about","above","after","again","against","ain","all","am","an","and","
            "as","at","be","because","been","before","being","below","between","both",
            "d","did","didn","didn't","do","does","doesn","doesn't","doing","don","don
            "for","from","further","had","hadn","hadn't","has","hasn","hasn't","have",
            "here","hers","herself","him","himself","his","how","i","if","in","into","
            "itself","just","ll","m","ma","me","mightn","mightn't","more","most","must
            "needn't","no","nor","not","now","o","of","off","on","once","only","or","o
            "out","over","own","re","s","same","shan","shan't","she","she's","should",
            "so","some","such","t","than","that","that'll","the","their","theirs","the
            "these","they","this","those","through","to","too","under","until","up","v
            "we","were","weren","weren't","what","when","where","which","while","who",
            "won't","wouldn","wouldn't","y","you","you'd","you'll","you're","you've","
            "yourselves","could","he'd","he'll","he's","here's","how's","i'd","i'll","
            "she'd","she'll","that's","there's","they'd","they'll","they're","they've"
            "what's","when's","where's","who's","why's","would","able","abst","accorda
            "across","act","actually","added","adj","affected","affecting","affects","
            "along","already","also","although","always","among","amongst","announce",
            "anymore","anyone","anything","anyway","anyways","anywhere","apparently","
            "around","aside","ask","asking","auth","available","away","awfully","b","b
            "becoming","beforehand","begin","beginning","beginnings","begins","behind"
            "beyond","biol","brief","briefly","c","ca","came","cannot","can't","cause"
            "co","com","come","comes","contain","containing","contains","couldnt","dat
            "due","e","ed","edu","effect","eg","eight","eighty","either","else","elsewh
            "especially","et","etc","even","ever","every","everybody","everyone","ever
            "f","far","ff","fifth","first","five","fix","followed","following","follow
            "found","four","furthermore","g","gave","get","gets","getting","give","giv
            "gone","got","gotten","h","happens","hardly","hed","hence","hereafter","he
            "hes","hi","hid","hither","home","howbeit","however","hundred","id","ie","
            "importance","important","inc","indeed","index","information","instead","i
            "it'll","j","k","keep","keeps","kept","kg","km","know","known","knows","l"
            "later","latter","latterly","least","less","lest","let","lets","like","lik
            "'ll","look","looking","looks","ltd","made","mainly","make","makes","many"
            "meantime","meanwhile","merely","mg","might","million","miss","ml","moreov
            "mug","must","n","na","name","namely","nay","nd","near","nearly","necessar
            "neither","never","nevertheless","new","next","nine","ninety","nobody","no
            "normally","nos","noted","nothing","nowhere","obtain","obtained","obviousl
            "omitted","one","ones","onto","ord","others","otherwise","outside","overal
            "particular","particularly","past","per","perhaps","placed","please","plus
            "potentially","pp","predominantly","present","previously","primarily","pro
            "provides","put","q","que","quickly","quite","qv","r","ran","rather","rd",
            "recently","ref","refs","regarding","regardless","regards","related","rela
            "resulted","resulting","results","right","run","said","saw","say","saying"
            "seeing","seem","seemed","seeming","seems","seen","self","selves","sent","
            "shes","show","showed","shown","showns","shows","significant","significant
            "six","slightly","somebody","somehow","someone","somethan","something","so
            "somewhere","soon","sorry","specifically","specified","specify","specifyin
            "sub","substantially","successfully","sufficiently","suggest","sup","sure"
            "tends","th","thank","thanks","thanx","thats","that've","thence","thereaft
            "therein","there'll","thereof","therere","theres","thereto","thereupon","t
            "thou","though","thoughh","thousand","throug","throughout","thru","thus","
            "toward","towards","tried","tries","truly","try","trying","ts","twice","tw
            "unless","unlike","unlikely","unto","upon","ups","us","use","used","useful
            "using","usually","v","value","various","'ve","via","viz","vol","vols","vs
            "wed","welcome","went","werent","whatever","what'll","whats","whence","whe
            "whereby","wherein","wheres","whereupon","wherever","whether","whim","whit
            "who'll","whomever","whos","whose","widely","willing","wish","within","wit
            "wouldnt","www","x","yes","yet","youd","youre","z","zero","a's","ain't","a
```

```
            "appreciate","appropriate","associated","best","better","c'mon","c's","can
            "consequently","consider","considering","corresponding","course","currentl
            "entirely","exactly","example","going","greetings","hello","help","hopeful
            "indicated","indicates","inner","insofar","it'd","keep","keeps","novel","p
            "secondly","sensible","serious","seriously","sure","t's","third","thorough
            "wonder"]
```

In [27]:

```
%time
# Combining all the above stundents
from tqdm import tqdm
preprocessed_project_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = 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_project_title.append(sent.lower().strip())
```

```
Wall time: 0 ns

100%|██████████████████████████████████████████████████████████████████████
███| 109248/109248 [00:14<00:00, 7460.99it/s]
```

In [28]:

```
preprocessed_project_title[99999]
```

Out[28]:

```
'turning flexible seating sixth grade class journey freedom'
```

1.3. Numerical normalization

1.3.1 normalization_price

In [29]:

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

Out[29]:

```
(109248, 20)
```

In [30]:

```
project_data.head(1)
```

Out[30]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Dat
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016 04-2 00:27:3

In [31]:

```
print(project_data["price"].shape)
```

```
(109248,)
```

In [32]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(1,-1)
price_normalize = Normalizer()
price_normalizer = price_normalize.fit_transform(project_data['price'].values.reshape(1
price_normalizer = price_normalizer.T
print(price_normalizer)
print("-------------------------------------------------------")
print("shape of price_normalizer:", price_normalizer.shape)
```

```
[[4.63560392e-03]
 [1.36200635e-03]
 [2.10346002e-03]
 ...
 [2.55100471e-03]
 [1.83960046e-03]
 [3.51642253e-05]]
-------------------------------------------------------
shape of price_normalizer: (109248, 1)
```

1.3.2 Normalization of teacher_number_of_previously_posted_projects

In [33]:

```
project_data['teacher_number_of_previously_posted_projects'].values
```

Out[33]:

```
array([53,  4, 10, ...,  0,  1,  2], dtype=int64)
```

In [34]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
teacher_number_of_previously_posted_projects_normalize = Normalizer()
teacher_number_of_previously_posted_projects_normalizer = teacher_number_of_previously_
teacher_number_of_previously_posted_projects_normalizer = teacher_number_of_previously_
print(teacher_number_of_previously_posted_projects_normalizer)
print("="*25)
print("Shape of teacher_number_of_previously_posted_projects_normalizer :", teacher_numb
```

```
[[0.00535705]
 [0.00040431]
 [0.00101076]
 ...
 [0.        ]
 [0.00010108]
 [0.00020215]]
=========================
Shape of teacher_number_of_previously_posted_projects_normalizer : (10924
8, 1)
```

1.3.3 spilt the data into train ,CV and test

In [35]:

```
project_data.head(1)
```

Out[35]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Dat
0	8393	p205479	2bf07ba08945e5d8b2a3f269b2b3cfe5	Mrs.	CA	2016 04-2 00:27:3

In [36]:

```
project_data['project_is_approved'].values
```

Out[36]:

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

In [37]:

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

In [38]:

```
# spliting the data into train , test CV
# Refrence link :https://scikit-learn.org/stable/modules/generated/sklearn.model_select
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data, label, test_size=0.33
X_train, X_cv,  y_train, y_cv    = train_test_split(X_train, y_train, test_size=0.33) #

print("Shape of X_train and y_train  :", X_train.shape, y_train.shape)
print("Shape of X_test and y_test    :", X_test.shape, y_test.shape)
print("Shape of X_cv and y_cv        :", X_cv.shape, y_cv.shape)
```

```
Shape of X_train and y_train  : (49041, 19) (49041,)
Shape of X_test and y_test    : (36052, 19) (36052,)
Shape of X_cv and y_cv        : (24155, 19) (24155,)
```

1.4. Vectorizing Categorical data

1.4.1 Vectorization of project_subject_categories¶

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

In [39]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=Fal

clean_categories_one_hot_train = vectorizer_cat.fit_transform(X_train['clean_categories
clean_categories_one_hot_test  = vectorizer_cat.transform(X_test['clean_categories'].va
clean_categories_one_hot_cv    = vectorizer_cat.transform(X_cv['clean_categories'].valu

print("vectorizer of project_subject_categories feature names :", vectorizer_cat.get_fe
print("-------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",clean_categories_one_hot_train.s
print("Shape of matrix after one hot encodig test  : ",clean_categories_one_hot_test.sh
print("Shape of matrix after one hot encodig cv    : ",clean_categories_one_hot_cv.shap
```

```
vectorizer of project_subject_categories feature names : ['Warmth', 'Care_
Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeed
s', 'Health_Sports', 'Math_Science', 'Literacy_Language']
-------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 9)
Shape of matrix after one hot encodig test  :  (36052, 9)
Shape of matrix after one hot encodig cv    :  (24155, 9)
```

1.4.2 vectorization of project_subject_subcategories

In [40]:

```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_subcat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowerc

clean_subcategories_one_hot_train = vectorizer_subcat.fit_transform(X_train['clean_subc
clean_subcategories_one_hot_test  = vectorizer_subcat.transform(X_test['clean_subcatego
clean_subcategories_one_hot_cv    = vectorizer_subcat.transform(X_cv['clean_subcategori

print("vectorizer of project_subject_subcategories feature names :", vectorizer_subcat.
print("-------------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",clean_subcategories_one_hot_trai
print("Shape of matrix after one hot encodig test  : ",clean_subcategories_one_hot_test
print("Shape of matrix after one hot encodig cv    : ",clean_subcategories_one_hot_cv.s
```

```
vectorizer of project_subject_subcategories feature names : ['Economics',
'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurric
ular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'War
mth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducati
on', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geogra
phy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'Env
ironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'S
pecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
-------------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 30)
Shape of matrix after one hot encodig test  :  (36052, 30)
Shape of matrix after one hot encodig cv    :  (24155, 30)
```

1.4.3 Vectorization of `school_state`

In [41]:

```
# we use count vectorizer to convert the values into one
vectorizer_ss = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lower

school_state_one_hot_train = vectorizer_ss.fit_transform(X_train['school_state'].values
school_state_one_hot_test  = vectorizer_ss.transform(X_test['school_state'].values)
school_state_one_hot_cv    = vectorizer_ss.transform(X_cv['school_state'].values)

print("vectorizer of school_state feature names :",vectorizer_ss.get_feature_names())
print("-------------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",school_state_one_hot_train.shape
print("Shape of matrix after one hot encodig test  : ",school_state_one_hot_test.shape)
print("Shape of matrix after one hot encodig cv    : ",school_state_one_hot_cv.shape)
```

```
vectorizer of school_state feature names : ['VT', 'WY', 'ND', 'MT', 'RI',
'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'IA', 'I
D', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'A
L', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'MA', 'LA', 'OH', 'MO', 'IN', 'P
A', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX', 'CA']
-------------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 51)
Shape of matrix after one hot encodig test  :  (36052, 51)
Shape of matrix after one hot encodig cv    :  (24155, 51)
```

1.4.4 Vectorization of `teacher_prefix`

In [42]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_tp = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict.keys()), low

teacher_prefix_one_hot_train = vectorizer_tp.fit_transform(X_train['teacher_prefix'].va
teacher_prefix_one_hot_test  = vectorizer_tp.transform(X_test['teacher_prefix'].values.
teacher_prefix_one_hot_cv    = vectorizer_tp.transform(X_cv['teacher_prefix'].values.as

print("Vectorizer of teacher_prefix :",vectorizer_tp.get_feature_names())
print("-------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",teacher_prefix_one_hot_train.sha
print("Shape of matrix after one hot encodig test  : ",teacher_prefix_one_hot_test.shap
print("Shape of matrix after one hot encodig cv    : ",teacher_prefix_one_hot_cv.shape)
```

```
Vectorizer of teacher_prefix : ['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
-------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 5)
Shape of matrix after one hot encodig test  :  (36052, 5)
Shape of matrix after one hot encodig cv    :  (24155, 5)
```

In [43]:

```
vectorizer_cat.get_feature_names
```

Out[43]:

```
<bound method CountVectorizer.get_feature_names of CountVectorizer(analyze
r='word', binary=True, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='cont
ent',
                lowercase=False, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None,
                vocabulary=['Warmth', 'Care_Hunger', 'History_Civics',
                            'Music_Arts', 'AppliedLearning', 'SpecialNeed
s',
                            'Health_Sports', 'Math_Science',
                            'Literacy_Language'])>
```

1.4.5 Vectorization of project_grade_categorie

In [44]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_pgc = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.key

project_grade_category_one_hot_train = vectorizer_pgc.fit_transform(X_train['project_gr
project_grade_category_one_hot_test  = vectorizer_pgc.transform(X_test['project_grade_c
project_grade_category_one_hot_cv    = vectorizer_pgc.transform(X_cv['project_grade_cat

print("Vectorizer of project_grade_categories :",vectorizer_pgc.get_feature_names())
print("-------------------------------------------------------")
print("Shape of matrix after one hot encodig train  : ",project_grade_category_one_hot_
print("Shape of matrix after one hot encodig test   : ",project_grade_category_one_hot_
print("Shape of matrix after one hot encodig cv     : ",project_grade_category_one_hot_
```

```
Vectorizer of project_grade_categories : ['Grades9-12', 'Grades6-8', 'Grad
es3-5', 'GradesPreK-2']
-------------------------------------------------------
Shape of matrix after one hot encodig train  :  (49041, 4)
Shape of matrix after one hot encodig test   :  (36052, 4)
Shape of matrix after one hot encodig cv     :  (24155, 4)
```

1.5. Vectorizing Text

1.5.1 Vectorization of essays bow

In [45]:

```
X_train['essay'].tail(1)
```

Out[45]:

```
68333    My students require technology in the form of ...
Name: essay, dtype: object
```

In [46]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_essays_bow = CountVectorizer(preprocessed_essays, min_df=10,  ngram_range=(1

essays_bow_train = vectorizer_essays_bow.fit_transform(X_train['essay'].values)
essays_bow_test  = vectorizer_essays_bow.transform(X_test['essay'].values)
essays_bow_cv    = vectorizer_essays_bow.transform(X_cv['essay'].values)

print("-------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",essays_bow_train.shape)
print("Shape of matrix after one hot encodig test  : ",essays_bow_test.shape)
print("Shape of matrix after one hot encodig cv    : ",essays_bow_cv.shape)
```

```
-------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 12571)
Shape of matrix after one hot encodig test  :  (36052, 12571)
Shape of matrix after one hot encodig cv    :  (24155, 12571)
```

In [47]:

```
print(vectorizer_essays_bow.get_feature_names)
```

```
<bound method CountVectorizer.get_feature_names of CountVectorizer(analyze
r='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8',
                input=['fortunate fairy tale stem kits classroom stem jour
nals '
                       'students enjoyed love implement lakeshore stem kit
s '
                       'classroom school year provide excellent engaging s
tem '
                       'lessons students variety backgrounds including '
                       'language socioeconomic status lot experience sc...
                       'requesting games practice phonics skills imagine r
ead '
                       'text presented clue read frustration intensified h
igh '
                       'stakes testing live support struggling readers lon
ger '
                       'struggling successful learners readers', ...],
                lowercase=True, max_df=1.0, max_features=None, min_df=10,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)>
```

1.5.1.1Vectorization of essay tfidf

In [48]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_essays_tfidf = TfidfVectorizer(preprocessed_essays, min_df=10, max_features=

essays_tfidf_train = vectorizer_essays_tfidf.fit_transform(X_train['essay'].values)
essays_tfidf_test  = vectorizer_essays_tfidf.transform(X_test['essay'].values)
essays_tfidf_cv    = vectorizer_essays_tfidf.transform(X_cv['essay'].values)

print("Shape of matrix after one hot encodig of train : ",essays_tfidf_train.shape)
print("Shape of matrix after one hot encodig test     : ",essays_tfidf_test.shape)
print("Shape of matrix after one hot encodig cv       : ",essays_tfidf_cv.shape)
```

```
Shape of matrix after one hot encodig of train :  (49041, 5000)
Shape of matrix after one hot encodig test     :  (36052, 5000)
Shape of matrix after one hot encodig cv       :  (24155, 5000)
```

1.5.1.2 Using Pretrained Models: essays Avg W2V

In [49]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[49]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/382303
```

```
49/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) load
GloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open
(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm
(f):\n        splitLine = line.split()\n        word = splitLine[0]\n
embedding = np.array([float(val) for val in splitLine[1:]])\n        mod
el[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n
return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ======
======================\nOutput:\n    \nLoading Glove Model\n1917495it [0
6:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ================
==========\n\nwords = []\nfor i in preproced_essays:\n    words.extend
(i.split(\' \'))\n\nfor i in preprocessed_project_title:\n    words.exte
nd(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwo
rds = set(words)\nprint("the unique words in the coupus", len(words))\n
\ninter_words = set(model.keys()).intersection(words)\nprint("The number
of words that are present in both glove vectors and our coupus",       l
en(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n
\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in words:\n
if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2
vec length", len(words_courpus))\n\n\n# stronging variables into pickle
 files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-
load-variables-in-python/\n\nimport (http://www.jessicayung.com/how-to-u
se-pickle-to-save-and-load-variables-in-python/\n\nimport) pickle\nwith
 open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus,
 f)\n\n\n'
```

In [50]:

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

In [51]:

```
# average Word2Vec X_train
# compute average word2vec for each review.
essays_avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is stored in
for sentence in tqdm(X_train['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    essays_avg_w2v_vectors_train.append(vector)

print(len(essays_avg_w2v_vectors_train))
print(len(essays_avg_w2v_vectors_train[0]))
```

```
100%|████████████████████████████████████████████████████████████████████
██████| 49041/49041 [00:55<00:00, 880.46it/s]

49041
300
```

average Word2Vec X_test_essay

In [52]:

```
# average Word2Vec X_test_essay
# compute average word2vec for each review.
essays_avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stored in t
for sentence in tqdm(X_test['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    essays_avg_w2v_vectors_test.append(vector)

print(len(essays_avg_w2v_vectors_test))
print(len(essays_avg_w2v_vectors_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
██████| 36052/36052 [00:42<00:00, 849.42it/s]

36052
300
```

average Word2Vec X_cv

In [53]:

```
# average Word2Vec X_cv
# compute average word2vec for each review.
essays_avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored in thi
for sentence in tqdm(X_cv['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    essays_avg_w2v_vectors_cv.append(vector)

print(len(essays_avg_w2v_vectors_cv))
print(len(essays_avg_w2v_vectors_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
██████| 24155/24155 [00:28<00:00, 857.76it/s]

24155
300
```

1.5.1.3 essays TFIDF weighted W2V train

In [54]:

```
tfidf_model_preprocessed_essays_train = TfidfVectorizer()
tfidf_model_preprocessed_essays_train.fit(X_train['essay'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_essays_train.get_feature_names(), list(t
tfidf_words = set(tfidf_model_preprocessed_essays_train.get_feature_names())
```

In [55]:

```
# essays TFIDF weighted W2V_train
# compute average word2vec for each review.
preprocessed_essays_train_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/revie
for sentence in tqdm(X_train['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_essays_train_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_essays_train_tfidf_w2v_vectors))
print(len(preprocessed_essays_train_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
███████| 49041/49041 [08:46<00:00, 93.08it/s]

49041
300
```

In [56]:

```
# tfidf_model_preprocessed_essays_test
# compute average word2vec for each review.
preprocessed_essays_test_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review
for sentence in tqdm(X_test['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_essays_test_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_essays_test_tfidf_w2v_vectors))
print(len(preprocessed_essays_test_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████
███████| 36052/36052 [06:29<00:00, 92.55it/s]

36052
300
```

In [57]:

```
# tfidf_model_preprocessed_essays_cv
# compute average word2vec for each review.
preprocessed_essays_cv_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review i
for sentence in tqdm(X_cv['essay']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_essays_cv_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_essays_cv_tfidf_w2v_vectors))
print(len(preprocessed_essays_cv_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████
███████| 24155/24155 [04:16<00:00, 94.01it/s]

24155
300
```

1.5.2 Vectorization of project_title bow train, test, cv

In [58]:

```
X_train['project_title'].head(1)
```

Out[58]:

```
90494    Quiet Please
Name: project_title, dtype: object
```

In [59]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_project_title_bow = CountVectorizer(preprocessed_project_title, min_df=10,ng

project_title_bow_train = vectorizer_project_title_bow.fit_transform(X_train['project_t
project_title_bow_test  = vectorizer_project_title_bow.transform(X_test['project_title'
project_title_bow_cv    = vectorizer_project_title_bow.transform(X_cv['project_title'].

print("------------------------------------------------------------")
print("Shape of matrix after one hot encodig train : ",project_title_bow_train.shape)
print("Shape of matrix after one hot encodig test  : ",project_title_bow_test.shape)
print("Shape of matrix after one hot encodig cv    : ",project_title_bow_cv.shape)
```

```
------------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 2101)
Shape of matrix after one hot encodig test  :  (36052, 2101)
Shape of matrix after one hot encodig cv    :  (24155, 2101)
```

1.5.2.1 Vectorization of project_title tfidf train, test. cv

In [60]:

```
# we are considering only the words which appeared in at least 10 documents (rows or pr
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_project_title_tfidf = TfidfVectorizer(preprocessed_project_title, min_df=10,

project_title_tfidf_train = vectorizer_project_title_tfidf.fit_transform(X_train['proje
project_title_tfidf_test  = vectorizer_project_title_tfidf.transform(X_test['project_ti
project_title_tfidf_cv    = vectorizer_project_title_tfidf.transform(X_cv['project_titl

print("Shape of matrix after one hot encodig of train : ",project_title_tfidf_train.sha
print("Shape of matrix after one hot encodig test     : ",project_title_tfidf_test.shap
print("Shape of matrix after one hot encodig cv       : ",project_title_tfidf_cv.shape)
```

```
Shape of matrix after one hot encodig of train :  (49041, 2101)
Shape of matrix after one hot encodig test     :  (36052, 2101)
Shape of matrix after one hot encodig cv       :  (24155, 2101)
```

1.5.2.2 Using Pretrained Models: project_title Avg W2V train

In [61]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[61]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ======================\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ======================\n\nwords = []\nfor i in preproced_essays:\n    words.extend(i.split(\' \'))\n\nfor i in preprocessed_project_title:\n    words.extend(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords = set(words)\nprint("the unique words in the coupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\nprint("The number of words that are present in both glove vectors and our coupus",       len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport (http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport) pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n\n'
```

In [62]:

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

In [63]:

```
# average Word2Vec  project_title_train
# compute average word2vec for each review.
project_title_avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is sto
for sentence in tqdm(X_train['project_title']): # 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
    project_title_avg_w2v_vectors_train.append(vector)

print(len(project_title_avg_w2v_vectors_train))
print(len(project_title_avg_w2v_vectors_train[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████
████| 49041/49041 [00:00<00:00, 57987.65it/s]

49041
300
```

In [64]:

```
# average Word2Vec  project_title_test
# compute average word2vec for each review.
project_title_avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stor
for sentence in tqdm(X_test['project_title']): # 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
    project_title_avg_w2v_vectors_test.append(vector)

print(len(project_title_avg_w2v_vectors_test))
print(len(project_title_avg_w2v_vectors_test[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
████| 36052/36052 [00:00<00:00, 45296.10it/s]

36052
300
```

In [65]:

```
# average Word2Vec  project_title_ cv
# compute average word2vec for each review.
project_title_avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(X_cv['project_title']): # 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
    project_title_avg_w2v_vectors_cv.append(vector)

print(len(project_title_avg_w2v_vectors_cv))
print(len(project_title_avg_w2v_vectors_cv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
████| 24155/24155 [00:00<00:00, 45347.82it/s]

24155
300
```

1.5.2.3 project_title TFIDF weighted W2V train

In [66]:

```
tfidf_model_preprocessed_project_title_train = TfidfVectorizer()
tfidf_model_preprocessed_project_title_train.fit(X_train['project_title'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model_preprocessed_project_title_train.get_feature_names(),
tfidf_words = set(tfidf_model_preprocessed_project_title_train.get_feature_names())
```

In [67]:

```
# project_title TFIDF weighted W2V train
# compute average word2vec for each review.
preprocessed_project_title_train_tfidf_w2v_vectors = []; # the avg-w2v for each sentenc
for sentence in tqdm(X_train['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_project_title_train_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_project_title_train_tfidf_w2v_vectors))
print(len(preprocessed_project_title_train_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████
████| 49041/49041 [00:01<00:00, 34944.92it/s]

49041
300
```

In [68]:

```
# project_title TFIDF weighted W2V_ test
# compute average word2vec for each review.
preprocessed_project_title_test_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/
for sentence in tqdm(X_test['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_project_title_test_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_project_title_test_tfidf_w2v_vectors))
print(len(preprocessed_project_title_test_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████
████| 36052/36052 [00:01<00:00, 31539.80it/s]

36052
300
```

In [69]:

```
# project_title TFIDF weighted W2V_cv
# compute average word2vec for each review.
preprocessed_project_title_cv_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/r
for sentence in tqdm(X_cv['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sen
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # ge
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    preprocessed_project_title_cv_tfidf_w2v_vectors.append(vector)

print(len(preprocessed_project_title_cv_tfidf_w2v_vectors))
print(len(preprocessed_project_title_cv_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████
████| 24155/24155 [00:00<00:00, 29999.77it/s]

24155
300
```

1.6. Vectorizing Numerical features

1.6.1 Normalization of price_train_test_cv

In [70]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(-1, 1)
price_normalizer = Normalizer()
price_normalizer_train = price_normalizer.fit_transform(X_train['price'].values.reshape
price_normalizer_test  = price_normalizer.transform(X_test['price'].values.reshape(1,-1
price_normalizer_cv    = price_normalizer.transform(X_cv['price'].values.reshape(1,-1))

# https://docs.scipy.org/doc/numpy/reference/generated/numpy.reshape.html
# Transpose the array
price_normalizer_train = price_normalizer_train.T
price_normalizer_test  = price_normalizer_test.T
price_normalizer_cv    = price_normalizer_cv.T

print("shape of price_normalizer_train:", price_normalizer_train.shape)
print("----------------------------")
print(price_normalizer_train)

print("shape of price_normalizer_test :", price_normalizer_test.shape)
print("----------------------------")
print(price_normalizer_test)

print("shape of price_normalizer_cv   :", price_normalizer_cv.shape)
print("----------------------------")
print(price_normalizer_cv)
```

```
shape of price_normalizer_train: (49041, 1)
----------------------------
[[6.12900562e-04]
 [6.45977735e-04]
 [2.72283507e-03]
 ...
 [5.90524827e-05]
 [2.82119102e-03]
 [1.78470807e-03]]
shape of price_normalizer_test : (36052, 1)
----------------------------
[[0.00578979]
 [0.00536041]
 [0.00295505]
 ...
 [0.00112864]
 [0.00113467]
 [0.00536988]]
shape of price_normalizer_cv   : (24155, 1)
----------------------------
[[0.00157102]
 [0.0088331 ]
 [0.00473456]
 ...
 [0.00132248]
 [0.00137762]
 [0.00769326]]
```

1.6.2 Teacher_number_of_previously_posted_projects_train_test_cv : Numerical / Normalization

Numerical : Normalization

In [71]:

```
# Link: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normali
from sklearn.preprocessing import Normalizer
# Reshaping price data using array.reshape(-1, 1)
teacher_number_of_previously_posted_projects_normalizer = Normalizer()

teacher_number_of_previously_posted_projects_normalizer_train = teacher_number_of_previ
teacher_number_of_previously_posted_projects_normalizer_test  = teacher_number_of_previ
teacher_number_of_previously_posted_projects_normalizer_cv    = teacher_number_of_previ

teacher_number_of_previously_posted_projects_normalizer_train=teacher_number_of_previou
teacher_number_of_previously_posted_projects_normalizer_test=teacher_number_of_previous
teacher_number_of_previously_posted_projects_normalizer_cv = teacher_number_of_previous

print("shape of teacher_number_of_previously_posted_projects_normalizer_train:",teacher_
print("----------------------------")
print(teacher_number_of_previously_posted_projects_normalizer_train)

print("shape of teacher_number_of_previously_posted_projects_normalizer_test :",teacher_
print("----------------------------")
print(teacher_number_of_previously_posted_projects_normalizer_test)

print("shape of teacher_number_of_previously_posted_projects_normalizer_cv   :",teacher_
print("----------------------------")
print(teacher_number_of_previously_posted_projects_normalizer_cv)
```

```
shape of teacher_number_of_previously_posted_projects_normalizer_train: (4
9041, 1)
----------------------------
[[0.00454089]
 [0.00030273]
 [0.00030273]
 ...
 [0.00045409]
 [0.        ]
 [0.00090818]]
shape of teacher_number_of_previously_posted_projects_normalizer_test : (3
6052, 1)
----------------------------
[[0.00247489]
 [0.        ]
 [0.        ]
 ...
 [0.00106067]
 [0.00053033]
 [0.00017678]]
shape of teacher_number_of_previously_posted_projects_normalizer_cv   : (2
4155, 1)
----------------------------
[[0.00233281]
 [0.00784674]
 [0.        ]
 ...
 [0.        ]
 [0.00106037]
 [0.01017955]]
```

In [72]:

```
project_data.columns
```

Out[72]:

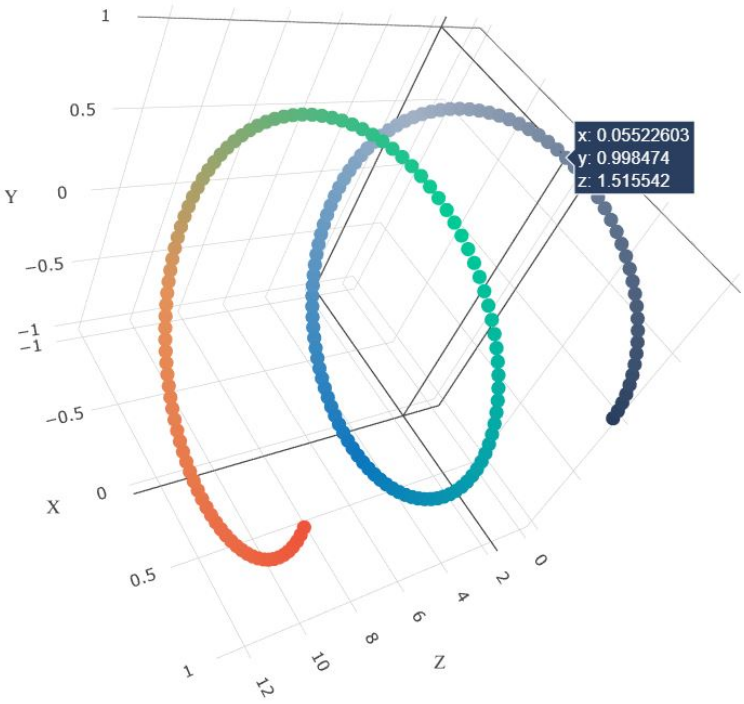
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_title', 'project_essay_
1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'clean_categories',
       'clean_subcategories', 'essay', 'price', 'quantity'],
      dtype='object')
```

we are going to consider

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

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

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

Assignment : DT

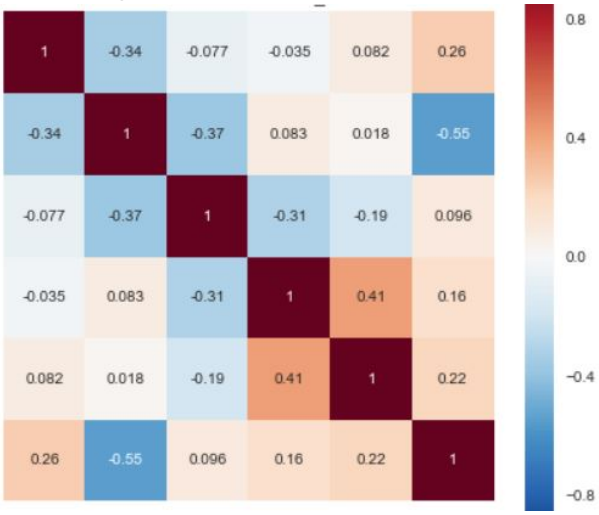
1. **Apply Decision Tree Classifier(DecisionTreeClassifier) on these feature sets**
    - Set 1: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
    - Set 2: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)
2. **The hyper paramter tuning (best `depth` in range [1, 5, 10, 50], and the best `min_samples_split` in range [5, 10, 100, 500])**
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - find the best hyper paramter using k-fold cross validation(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*

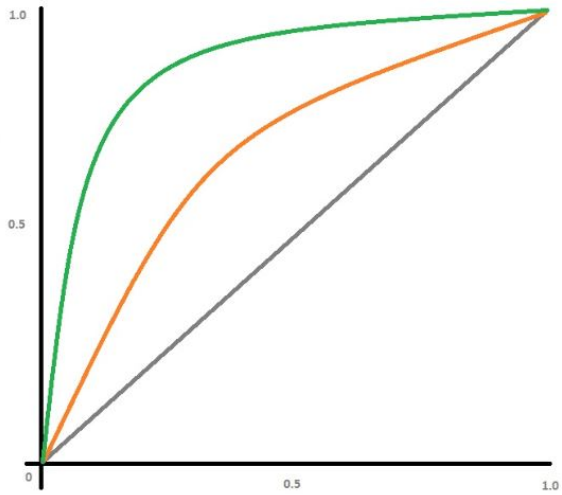
or

- 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 (https://seaborn.pydata.org/generated/seaborn.heatmap.html) with rows as **n_estimators**, columns as **max_depth**, and values inside the cell representing **AUC Score**

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



- Along with plotting ROC curve, you need to print the [confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/)](https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points

	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/ (https://www.geeksforgeeks.org/generating-word-cloud-python/)](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) (https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html))](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

2. Decision Tree

2.1 Applying Decision Tree on BOW, SET 1 - Experiment

Merging features encoding numerical + categorical features BOW, SET 1

In [73]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set1_ = all categorical features + numarical features + project_title(BOW) + preproce

set1_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     essays_bow_train,teacher_number_of_previously_posted_projects_norm
                     price_normalizer_train)).tocsr()

set1_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, sc
                    teacher_prefix_one_hot_test,project_grade_category_one_hot_test, pr
                    essays_bow_test,teacher_number_of_previously_posted_projects_normal
                    price_normalizer_test)).tocsr()

set1_cv  = hstack((clean_categories_one_hot_cv, clean_subcategories_one_hot_cv,school_s
                   teacher_prefix_one_hot_cv,project_grade_category_one_hot_cv,project_
                   essays_bow_cv,teacher_number_of_previously_posted_projects_normalize
                   price_normalizer_cv)).tocsr()

print("Final Data Matrix of set1 :")
print("shape of set1_train and y_train :", set1_train.shape , y_train.shape)
print("shape of set1_test and y_test   :", set1_test.shape , y_test.shape)
print("shape of set1_cv and y_cv       :", set1_cv.shape , y_cv.shape)
```

```
Final Data Matrix of set1 :
shape of set1_train and y_train : (49041, 14773) (49041,)
shape of set1_test and y_test   : (36052, 14773) (36052,)
shape of set1_cv and y_cv       : (24155, 14773) (24155,)
```

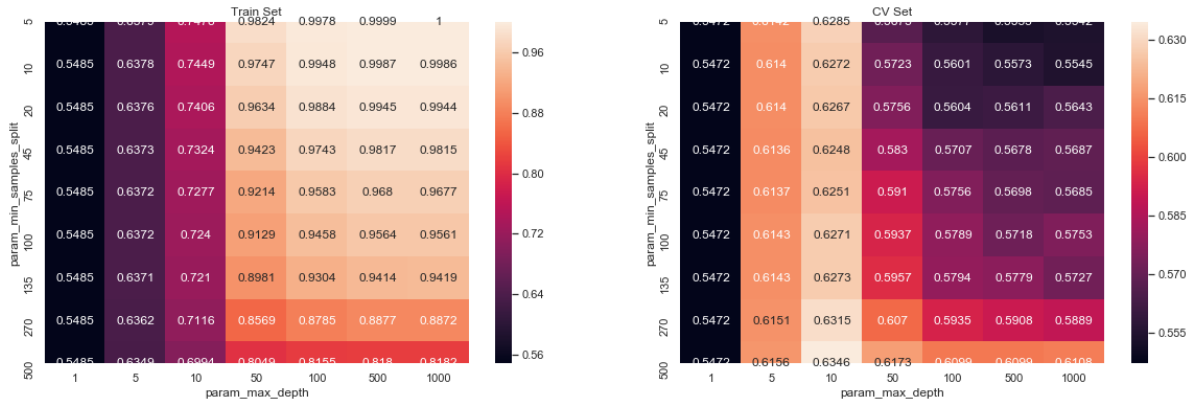
2.1.1 Hyper parameter Tuning to find best estimator :: GridSearchCV set1_Bow (Experiment)

In [74]:

```
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
dt1 = DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50, 100, 500, 1000], 'min_samples_split': [5, 10,
clf1 = GridSearchCV(dt1, parameters, cv=3, scoring='roc_auc',return_train_score=True)
se1 = clf1.fit(set1_train, y_train)
```

In [75]:

```
# ploting the performance of model both on train data and cross validation data for eac
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(clf1.cv_results_).groupby(['param_min_samples_split', 'param

fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [75]:

```
# Best tune parameters with score
print(clf1.best_estimator_)
print(clf1.score(set1_train,y_train))
print(clf1.score(set1_test,y_test))
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_dept
h=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')
0.7026895650424716
0.6421025046804463
```

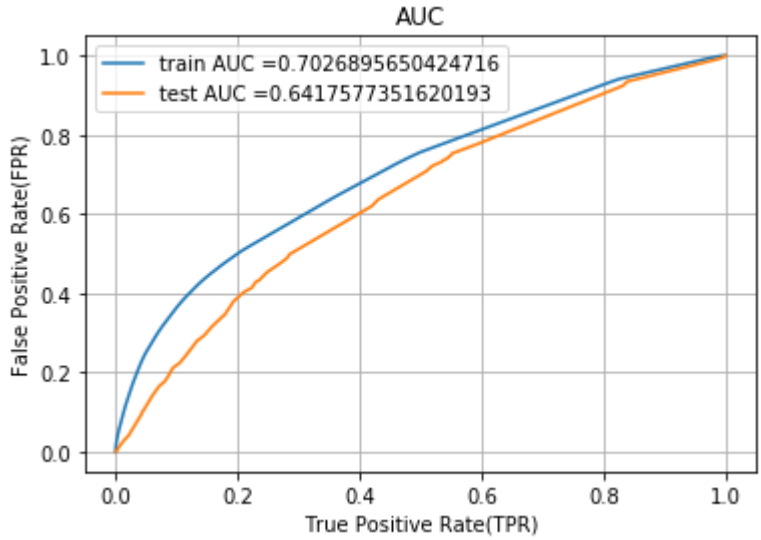
In [76]:

```
# best tune parameters
best_tune_parameters=[{'max_depth':[10], 'min_samples_split':[500]}]
```

2.1.2.Train model using the best hyper-parameter value set1_bow

In [77]:

```
%%time
from sklearn.metrics import auc,roc_curve

clf_1v =DecisionTreeClassifier (class_weight = 'balanced',max_depth=10,min_samples_spli
clf_1v.fit(set1_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_1v.predict_proba(set1_train)[:
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_1v.predict_proba(set1_test)[:,1]

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.grid(True)
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.show()
```



Wall time: 14.2 s

2.1.3. Confustion Matrix set1_train and set1_test

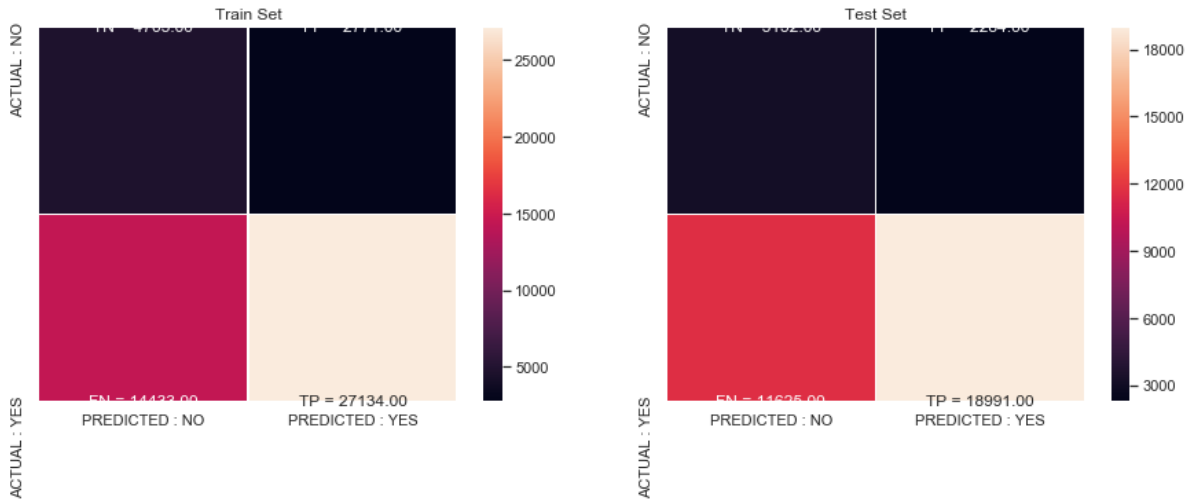
In [78]:

```
def predict(proba, threshould, fpr, tpr):
    t = threshould[np.argmax(fpr*(1-tpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.rou
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:predictions.append(0)
    return predictions
```

In [79]:

```
clf_1v.fit(set1_train,y_train)
y_train_pred_1 = clf_1v.predict_proba(set1_train)[:,1]
y_test_pred_1 = clf_1v.predict_proba(set1_test)[:,1]


#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
conf_matr_df_train_1 = confusion_matrix(y_train, predict(y_train_pred_1, thresholds, tr
conf_matr_df_test_1 = confusion_matrix(y_test, predict(y_test_pred_1, thresholds, test_

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k

labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(ke

sns.heatmap(conf_matr_df_train_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''

sns.heatmap(conf_matr_df_test_1, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.41283659907058556 for threshold 0.454
the maximum value of tpr*(1-fpr) 0.3623010209496716 for threshold 0.471
```



2.1.4 Graphviz visualization of Decision Tree on BOW, SET 1

In [80]:

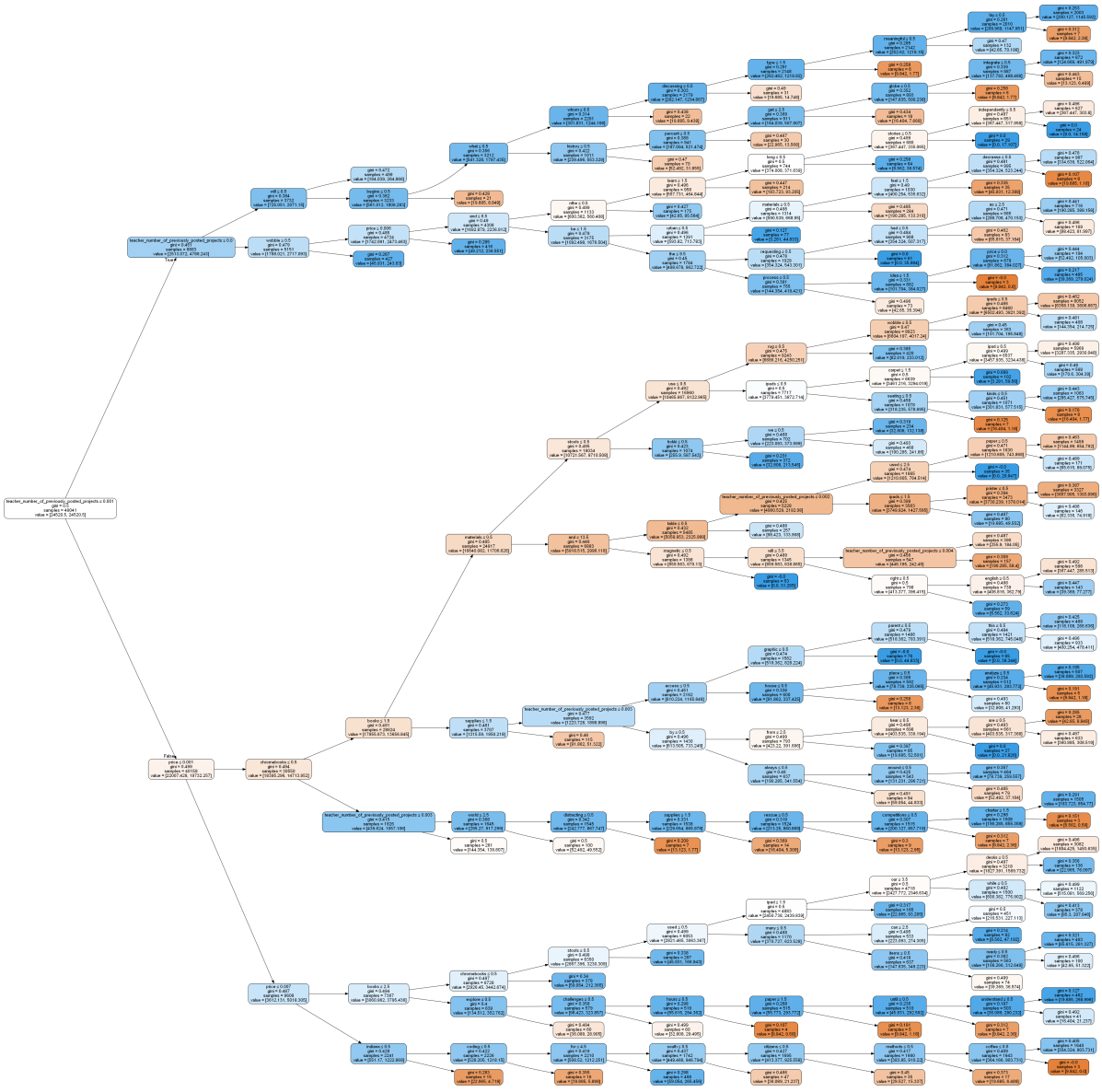
```
# feature aggregation

f1 = vectorizer_cat.get_feature_names()
f2 = vectorizer_subcat.get_feature_names()
f3 = vectorizer_ss.get_feature_names()
f4 = vectorizer_pgc.get_feature_names()
f5 = vectorizer_tp.get_feature_names()
f6 = vectorizer_project_title_bow.get_feature_names()
f7 = vectorizer_essays_bow.get_feature_names()
f8 = vectorizer_project_title_tfidf.get_feature_names()
f9 = vectorizer_essays_tfidf.get_feature_names()

feature_agg_bow = f1+f2+f3+f4+f5+f6+f7
feature_agg_tfidf = f1+f2+f3+f4+f5+f8+f9
feature_agg_bow.append('price')
feature_agg_bow.append('teacher_number_of_previously_posted_projects')
feature_agg_tfidf.append('price')
feature_agg_tfidf.append('teacher_number_of_previously_posted_projects')
```

In [81]:

```
#Link:https://datascience.stackexchange.com/questions/37428/graphviz-not-working-when-i
import os
os.environ['PATH'] = os.environ['PATH']+';'+os.environ['CONDA_PREFIX']+r"\Library\bin\g
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf_1v, out_file=dot_data, filled=True, rounded=True, special_character
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\externals\six.py:31: De
precationWarning: The module is deprecated in version 0.21 and will be rem
oved in version 0.23 since we've dropped support for Python 2.7. Please re
ly on the official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", DeprecationWarning)
```

Out[81]:



2.1.5 False Possitive points

In [82]:

```
# Confusion Matrix of Test
conf_matr_df_test_1
```

Out[82]:

```
array([[ 3152,  2284],
       [11625, 18991]], dtype=int64)
```

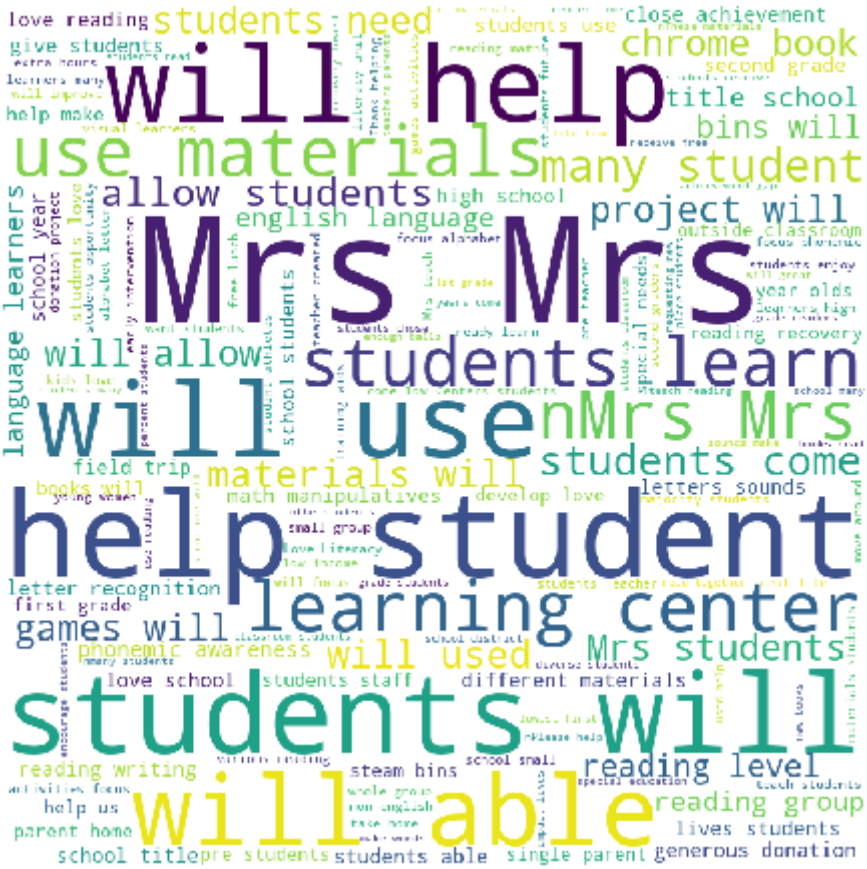
In [83]:

```
#Cite : https://datascience.stackexchange.com/questions/28493/confusion-matrix-get-item
fp =[]
for i in range(len(y_test)):
    if (y_test.values[i] == 0) & (y_test_pred_1[i] == 1) :
        fp.append(i)
fp_essay_1=[]
for i in fp:
    fp_essay_1.append(X_test['essay'].values[i])
```

2.1.6 Ploting the WordCloud with the words of essay text of these `false positive data points`

In [84]:

```
# Cite : https://www.geeksforgeeks.org/generating-word-cloud-python/
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay_1 : # iterate through the csv file
    val = str(val) # typecaste each val to string
    tokens = val.split() # split the value
    for i in range(len(tokens)): # Converts each token into lowercase
        tokens[i] = tokens[i].lower()
        for words in tokens :
            comment_words = comment_words + words + ' '
wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords =
# plot the WordCloud image
plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



2.1.7 Ploting Box plot with `price` of false possitive points

In [85]:

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

X_test_fp.head(1)
len(X_test_fp)
```

Out[85]:

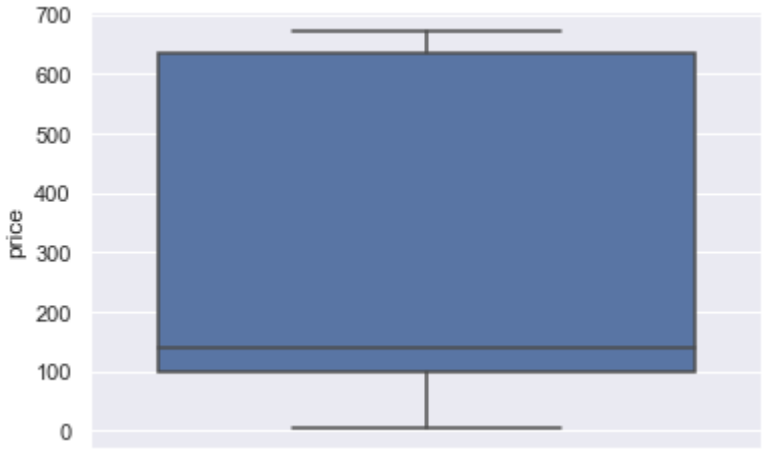
```
9
```

In [86]:

```
##Box Plot (FP 'price')
sns.boxplot(y='price', data=X_test_fp)
```

Out[86]:

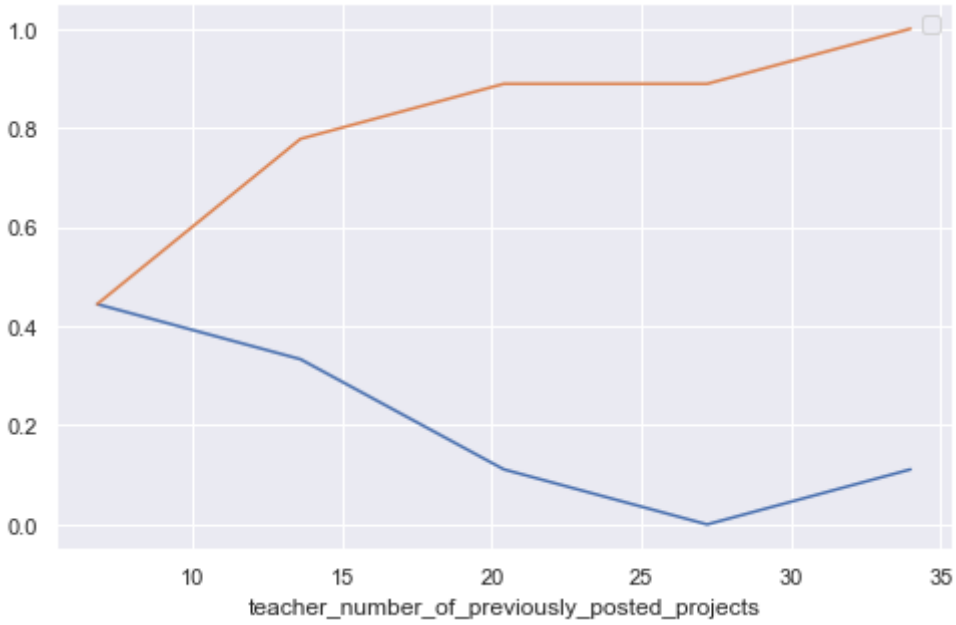
```
<matplotlib.axes._subplots.AxesSubplot at 0x2ae62594bc8>
```



2.1.8 PDF & CDF with `teacher_number_of_previously_posted_projects` false possitive points

In [87]:

```
plt.figure(figsize=(8,5))
counts, bin_edges = np.histogram(X_test_fp['teacher_number_of_previously_posted_project
pdf = counts/sum(counts)
cdf=np.cumsum(pdf)
pdf_plot = plt.plot(bin_edges[1:],pdf)
cdf_plot = plt.plot(bin_edges[1:],cdf)
plt.legend([pdf_plot, cdf_plot], ["PDF", "CDF"])
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



2.2 Applying Decision Trees on Tfidf, SET 2

In [88]:

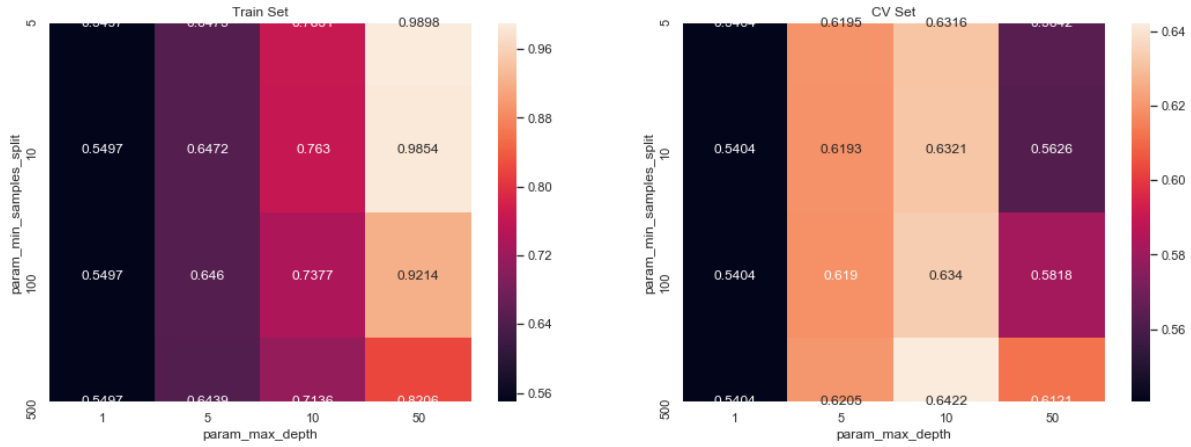
```
#merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set2_ = all categorical features + numarical features + essays_tfidf + project_title_

set2_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     essays_tfidf_train,teacher_number_of_previously_posted_projects_no
                     price_normalizer_train)).tocsr()


set2_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, sc
                    teacher_prefix_one_hot_test,project_grade_category_one_hot_test, pr
                    essays_tfidf_test,teacher_number_of_previously_posted_projects_norm
                    price_normalizer_test)).tocsr()

set2_cv  = hstack((clean_categories_one_hot_cv, clean_subcategories_one_hot_cv,school_s
                   teacher_prefix_one_hot_cv,project_grade_category_one_hot_cv,project_
                   essays_tfidf_cv,teacher_number_of_previously_posted_projects_normali
                   price_normalizer_cv)).tocsr()

print("Final Data Matrix of set2 :")
print("shape of set2_train and y_train :", set2_train.shape , y_train.shape)
print("shape of set2_test and y_test   :", set2_test.shape , y_test.shape)
print("shape of set2_cv and y_cv       :", set2_cv.shape , y_cv.shape)
```

```
Final Data Matrix of set2 :
shape of set2_train and y_train : (49041, 7202) (49041,)
shape of set2_test and y_test   : (36052, 7202) (36052,)
shape of set2_cv and y_cv       : (24155, 7202) (24155,)
```

2.2.1 Hyper parameter Tuning to find best estimator ::GridSearchCV set2_TFIDF

In [89]:

```
dt2 = DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10,100, 500]}
clf2 = GridSearchCV(dt2, parameters, cv=3, scoring='roc_auc',return_train_score=True)
set2_fit = clf2.fit(set2_train, y_train)

import seaborn as sns; sns.set()
max_scores2 = pd.DataFrame(clf2.cv_results_).groupby(['param_min_samples_split', 'param

fig, ax = plt.subplots(1,2, figsize=(18,6))
sns.heatmap(max_scores2.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores2.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [90]:

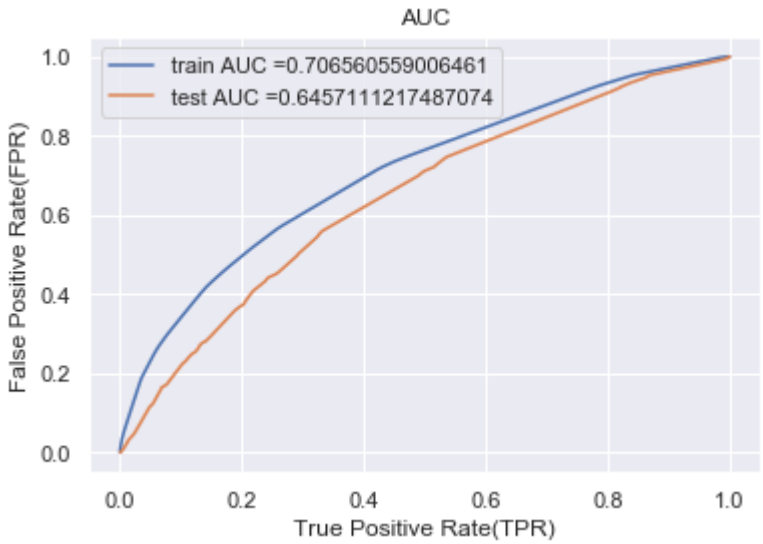
```
#Best Estimator and Best tune parameters
print(clf2.best_estimator_)
#Mean cross-validated score of the best_estimator
print(clf2.score(set2_train,y_train))
print(clf2.score(set2_test,y_test))
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_dept
h=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')
0.7065067707248756
0.6457383346234964
```

2.2.2 Train model using the best hyper-parameter set2(TFIDF)

In [91]:

```
%%time
from sklearn.metrics import auc,roc_curve

clf_2v =DecisionTreeClassifier (class_weight = 'balanced',max_depth=10,min_samples_spli
clf_2v.fit(set2_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_2v.predict_proba(set2_train)[:
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_2v.predict_proba(set2_test)[:,1]

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.grid(True)
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.show()
```



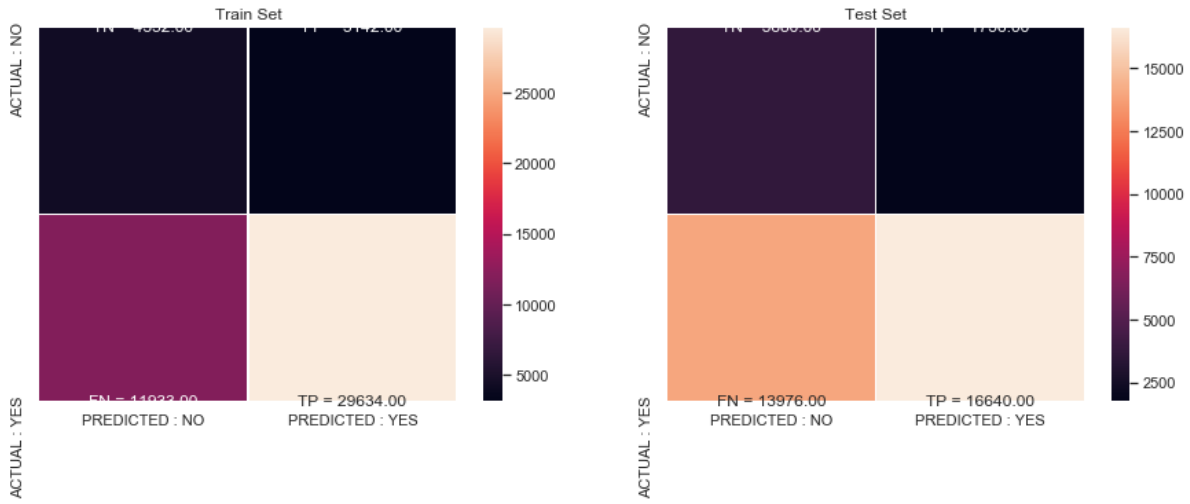

```
Wall time: 22 s
```

2.2.3 Confusion Matrix of set2_train , set2_test, TFIDF

In [92]:

```
clf_2v.fit(set2_train,y_train)
y_train_pred_2 = clf_2v.predict_proba(set2_train)[:,1]
y_test_pred_2 = clf_2v.predict_proba(set2_test)[:,1]


#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
conf_matr_df_train_2 = confusion_matrix(y_train, predict(y_train_pred_2, thresholds, tra
conf_matr_df_test_2 = confusion_matrix(y_test, predict(y_test_pred_2, thresholds, test_

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k

labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(ke

sns.heatmap(conf_matr_df_train_2, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''

sns.heatmap(conf_matr_df_test_2, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTI
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.4206873931553186 for threshold 0.475
the maximum value of tpr*(1-fpr) 0.37384660432352673 for threshold 0.537
```



In [93]:

```
print("Confusion Matrix of set2_train of TFIDF:",conf_matr_df_train_2)
print("Confusion Matrix of set2_test of TFIDF:",conf_matr_df_test_2)
```

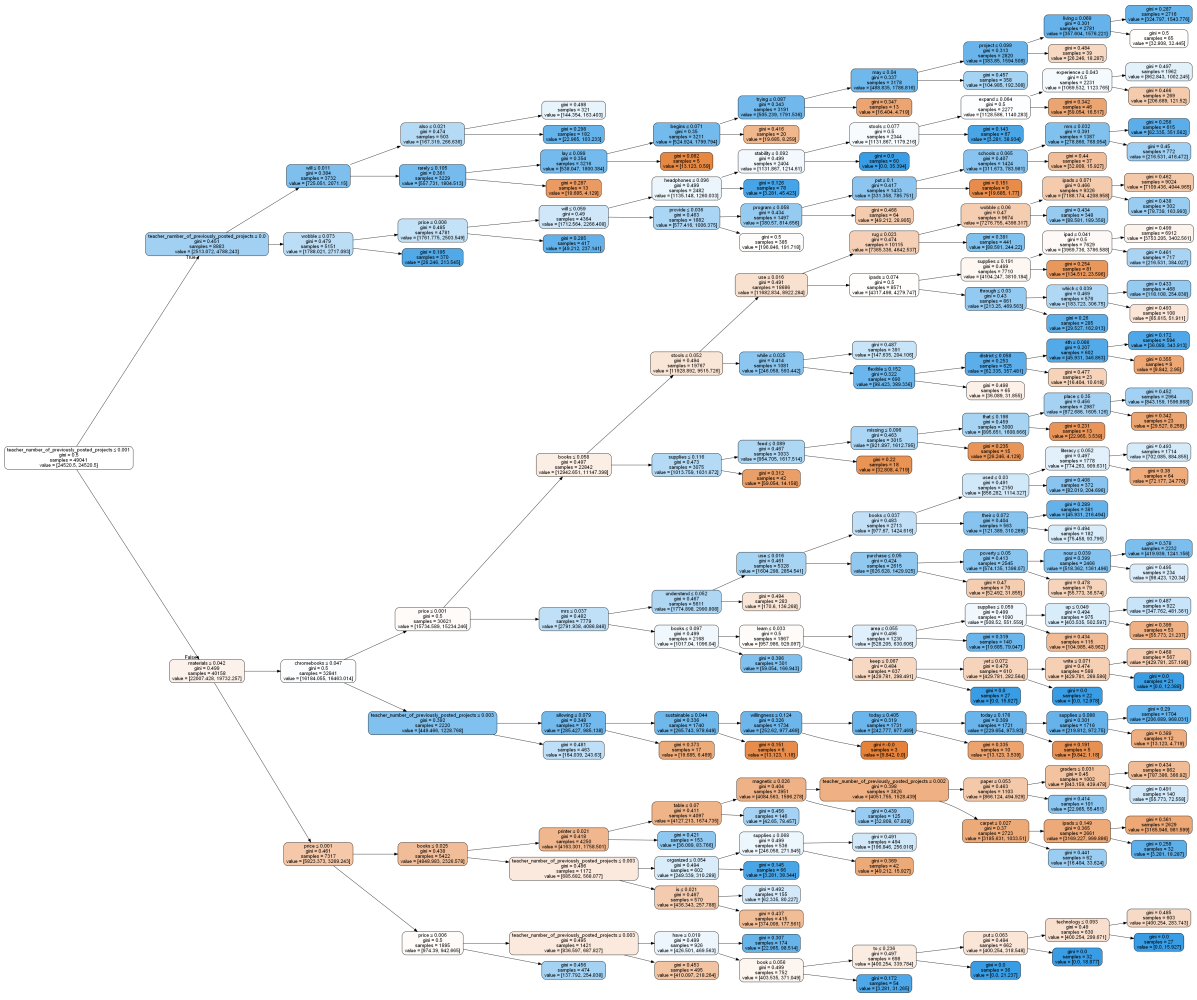
```
Confusion Matrix of set2_train of TFIDF: [[ 4332  3142]
 [11933 29634]]
Confusion Matrix of set2_test of TFIDF: [[ 3680  1756]
 [13976 16640]]
```

2.2.4 Graphviz visualization of Decision Tree on set2 TFIDF

In [94]:

```
#Link:https://datascience.stackexchange.com/questions/37428/graphviz-not-working-when-i
import os
os.environ['PATH'] = os.environ['PATH']+';'+os.environ['CONDA_PREFIX']+r"\Library\bin\g
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf_2v, out_file=dot_data, filled=True, rounded=True, special_character
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```

Out[94]:



2.2.5 False Possitive Points

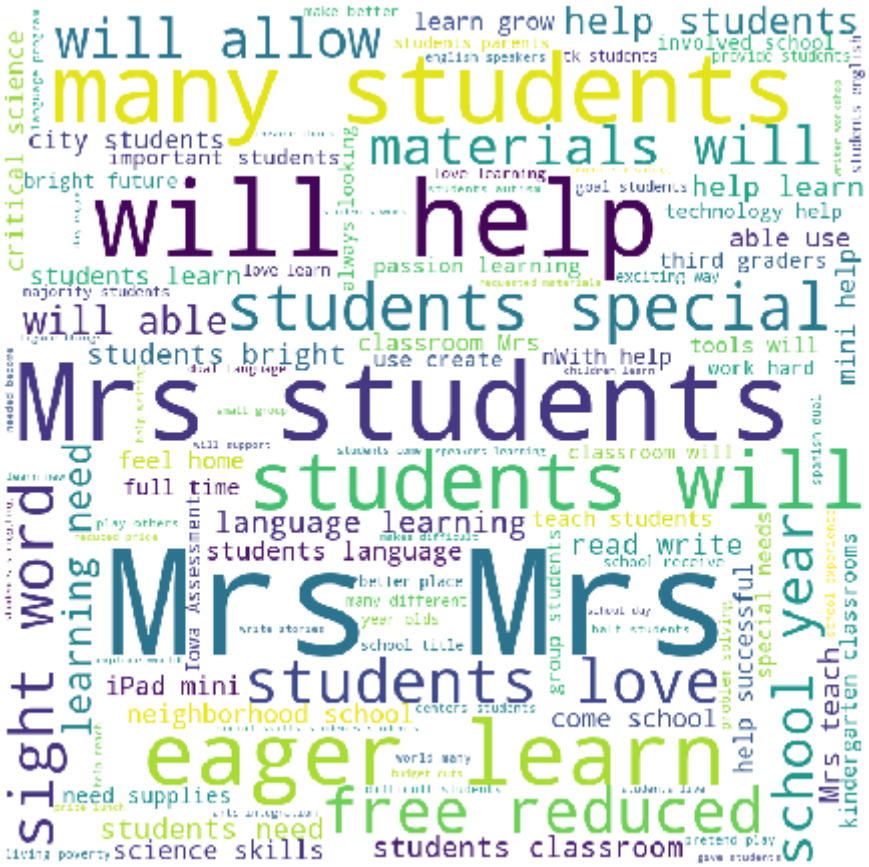
In [95]:

```
#Cite : https://datascience.stackexchange.com/questions/28493/confusion-matrix-get-item
fp =[]
for i in range(len(y_test)):
    if (y_test.values[i] == 0) & (y_test_pred_2[i] == 1) :
        fp.append(i)
fp_essay_2=[]
for i in fp:
    fp_essay_2.append(X_test['essay'].values[i])
```

2.1.6 Ploting the WordCloud with the words of `essay` text of these `false positive data points`

In [96]:

```
# Cite : https://www.geeksforgeeks.org/generating-word-cloud-python/
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay_2 : # iterate through the csv file
    val = str(val) # typecaste each val to string
    tokens = val.split() # split the value
    for i in range(len(tokens)): # Converts each token into lowercase
        tokens[i] = tokens[i].lower()
        for words in tokens :
            comment_words = comment_words + words + ' '
wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords =
# plot the WordCloud image
plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



2.1.7 Ploting Box plot with `price` of false possitive points

In [97]:

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

X_test_fp.head(1)
len(X_test_fp)
```

Out[97]:

```
6
```

In [98]:

```
##Box Plot (FP 'price')
sns.boxplot(y='price', data=X_test_fp)
```

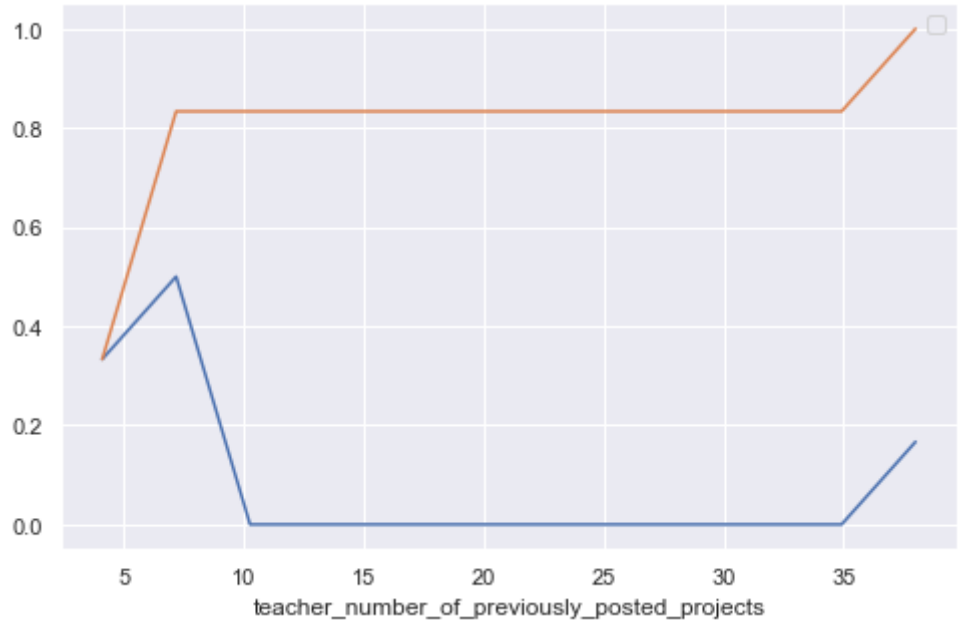
Out[98]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2ae5e786608>
```

2.2.8 PDF & CDF with teacher_number_of_previously_posted_projects of false possitive points

In [99]:

```
plt.figure(figsize=(8,5))
counts, bin_edges = np.histogram(X_test_fp['teacher_number_of_previously_posted_project
pdf = counts/sum(counts)
cdf=np.cumsum(pdf)
pdf_plot = plt.plot(bin_edges[1:],pdf)
cdf_plot = plt.plot(bin_edges[1:],cdf)
plt.legend([pdf_plot, cdf_plot], ["PDF", "CDF"])
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



2.3 Applying Decision Trees on AVG W2V, SET 3 Experiment

In [100]:

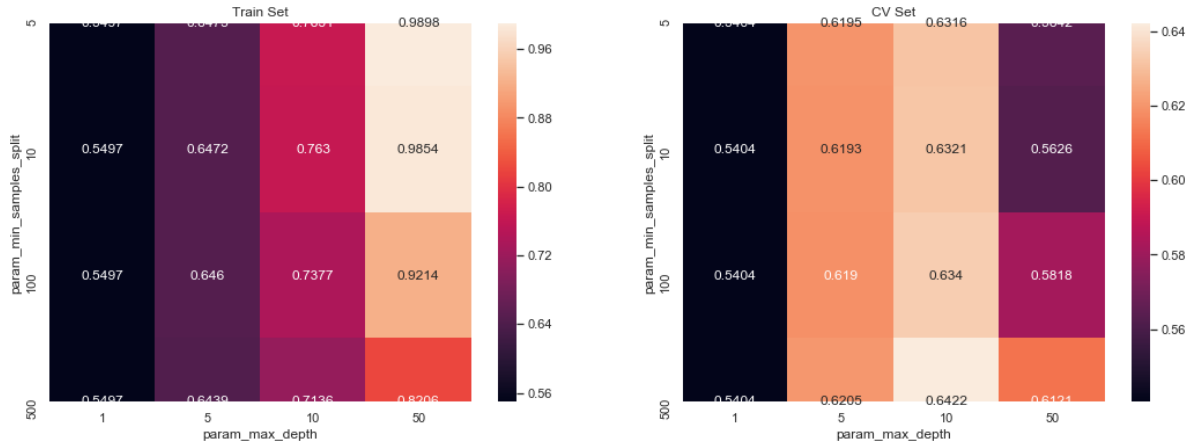
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set3_ = all categorical features + numarical features + essays_avg_w2v_vectors , proj


set3_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     project_title_avg_w2v_vectors_train,teacher_number_of_previously_p
                     price_normalizer_train))

set3_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, sc
                     teacher_prefix_one_hot_test,project_grade_category_one_hot_test, e
                     project_title_avg_w2v_vectors_test,teacher_number_of_previously_po
                     price_normalizer_test))

set3_cv = hstack((clean_categories_one_hot_cv, clean_subcategories_one_hot_cv, school_s
                     teacher_prefix_one_hot_cv,project_grade_category_one_hot_cv, essay
                     project_title_avg_w2v_vectors_cv,teacher_number_of_previously_post
                     price_normalizer_cv))

print("Final Data Matrix of set3 :")
print("shape of set3_train and y_train :", set3_train.shape , y_train.shape)
print("shape of set3_test and y_test   :", set3_test.shape , y_test.shape)
print("shape of set3_cv and y_cv       :", set3_cv.shape , y_cv.shape)
```

```
Final Data Matrix of set3 :
shape of set3_train and y_train : (49041, 701) (49041,)
shape of set3_test and y_test   : (36052, 701) (36052,)
shape of set3_cv and y_cv       : (24155, 701) (24155,)
```

2.3.1 Hyper parameter Tuning to find best estimator :: GridSearchCV `set3 (AVG W2V)`

In [101]:

```
dt3 = DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10,100, 500]}
clf3 = GridSearchCV(dt3, parameters, cv=5, scoring='roc_auc',return_train_score=True)
set3_fit = clf3.fit(set3_train, y_train)

import seaborn as sns; sns.set()
max_scores3 = pd.DataFrame(clf3.cv_results_).groupby(['param_min_samples_split', 'param_

fig, ax = plt.subplots(1,2, figsize=(18,6))
sns.heatmap(max_scores2.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores2.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [102]:

```
print(clf3.best_estimator_)
print(clf3.score(set3_train,y_train))
print(clf3.score(set3_test,y_test))
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_dept
h=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')
0.6467557923305022
0.6126780745873833
```

2.3.2 Train model using best hyper parameter values : set3 (AVG W2V)

In [103]:

```
%%time
from sklearn.metrics import auc,roc_curve

clf_3v =DecisionTreeClassifier (class_weight = 'balanced',max_depth=5,min_samples_split
clf_3v.fit(set3_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs

train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_3v.predict_proba(set3_train)[:
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_3v.predict_proba(set3_test)[:,1]

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))

plt.legend()
plt.grid(True)
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.show()
```

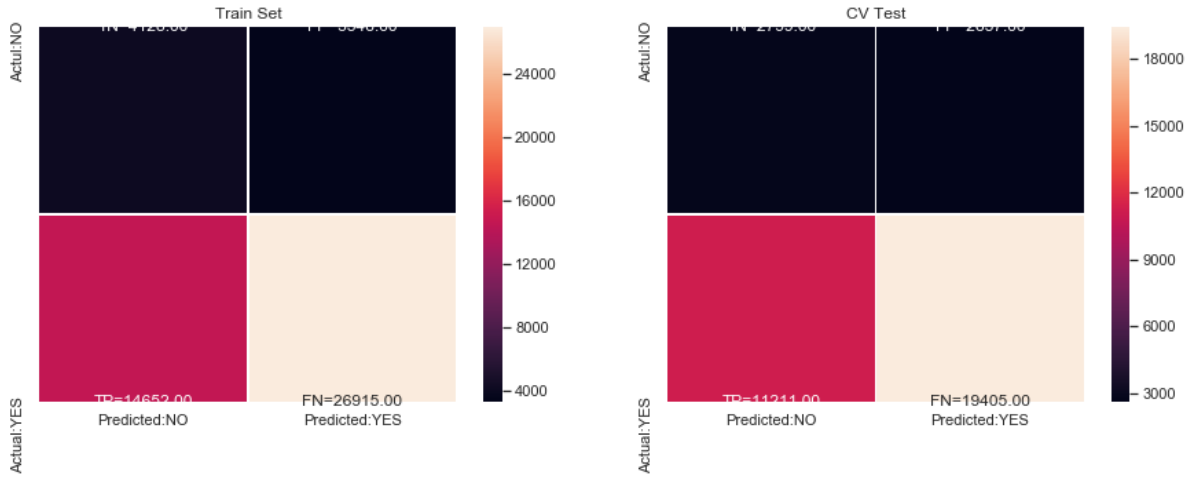

```
Wall time: 20.7 s
```

2.3.3 Confusion Matrix : set3_train, set3_test (AVG W2V)

In [104]:

```
clf_3v.fit(set3_train,y_train)
y_train_pred_3 = clf_3v.predict_proba(set3_train)[:,1]
y_test_pred_3 = clf_3v.predict_proba(set3_test)[:,1]

conf_matr_df_train_3 = confusion_matrix(y_train,predict(y_train_pred_3,thresholds,train_
conf_matr_df_test_3  = confusion_matrix(y_test, predict(y_test_pred_3, thresholds, test_

key=(np.asarray([['TN','FP'],['TP','FN']]))
fig,ax=plt.subplots(1,2,figsize=(15,5))
labels_train = np.asarray(['{0}={1:.2f}'.format(key,value) for key, value in zip(key.fl
labels_test  = np.asarray(['{0}={1:.2f}'.format(key,value) for key ,value in zip(key.fl

sns.heatmap(conf_matr_df_train_3, linewidths=.5, xticklabels=['Predicted:NO','Predicted
sns.heatmap(conf_matr_df_test_3,  linewidths=.5, xticklabels=['Predicted:NO','Predicted
ax[0].set_title('Train Set')
ax[1].set_title("CV Test")
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.35762864547217715 for threshold 0.489
the maximum value of tpr*(1-fpr) 0.3333294878398767 for threshold 0.489
```



2.3.4 False Possitive Points

In [107]:

```
conf_matr_df_test_3
```

Out[107]:

```
array([[ 2799,  2637],
       [11211, 19405]], dtype=int64)
```

In [115]:

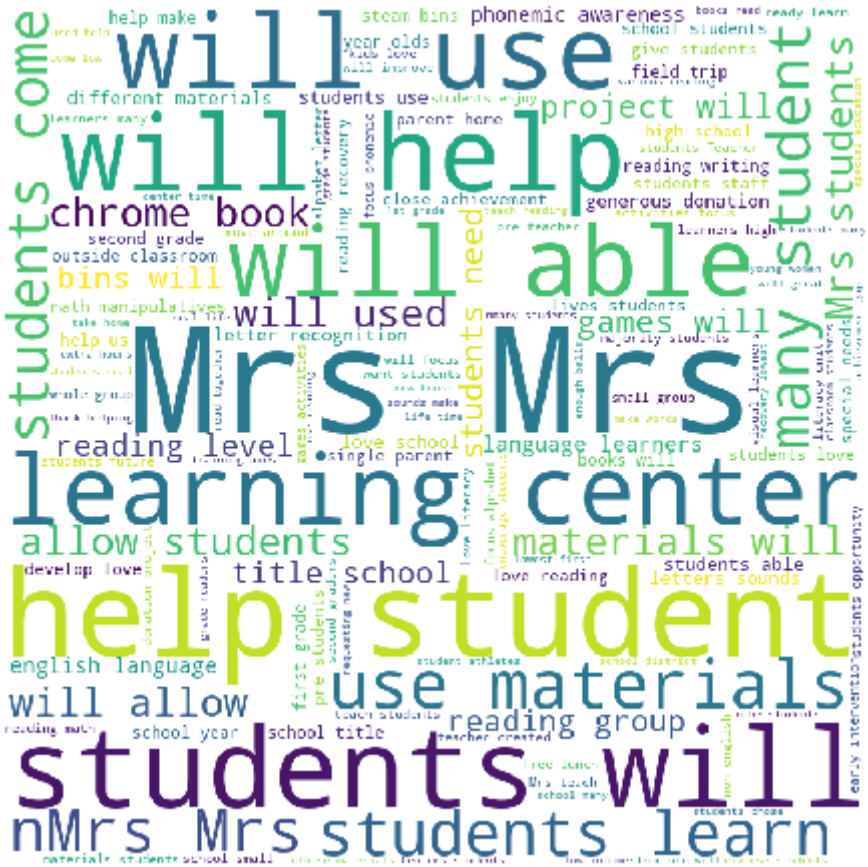
```
#Cite : https://datascience.stackexchange.com/questions/28493/confusion-matrix-get-item
fp =[]
for i in range(len(y_test)):
    if (y_test.values[i] == 0) & (y_test_pred_1[i] == 1) :
        fp.append(i)
fp_essay_3=[]
for i in fp:
    fp_essay_3.append(X_test['essay'].values[i])
print("False Possitive points:", fp)
```

```
False Possitive points: [305, 1595, 5589, 6059, 7263, 7382, 7652, 12932, 1
8745, 19879, 22383, 26416, 26889, 29645, 30092, 31006, 32229, 32641, 3351
9]
```

2.3.5 Ploting the WordCloud with the words of `essay text of these false positive` data points

In [116]:

```
# Cite : https://www.geeksforgeeks.org/generating-word-cloud-python/
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay_3 : # iterate through the csv file
    val = str(val) # typecaste each val to string
    tokens = val.split() # split the value
    for i in range(len(tokens)): # Converts each token into lowercase
        tokens[i] = tokens[i].lower()
        for words in tokens :
            comment_words = comment_words + words + ' '
wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords =
# plot the WordCloud image
plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



2.3.6 Ploting Box plot with `price of false possitive points`

In [117]:

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

X_test_fp.head(1)
len(X_test_fp)
```

Out[117]:

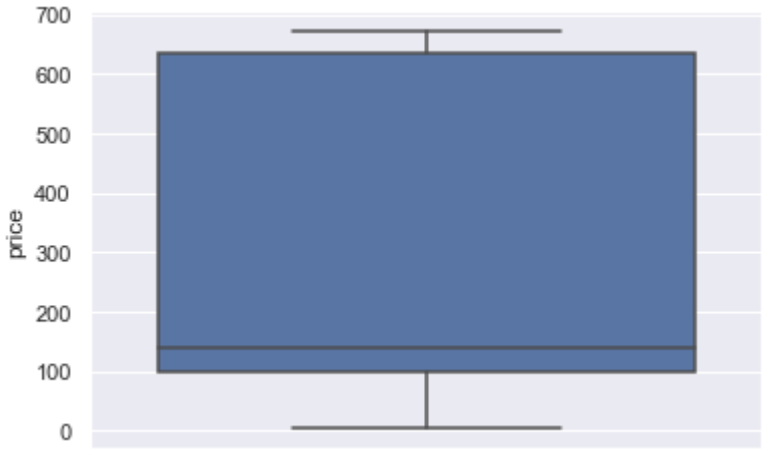
```
9
```

In [118]:

```
##Box Plot (FP 'price')
sns.boxplot(y='price', data=X_test_fp)
```

Out[118]:

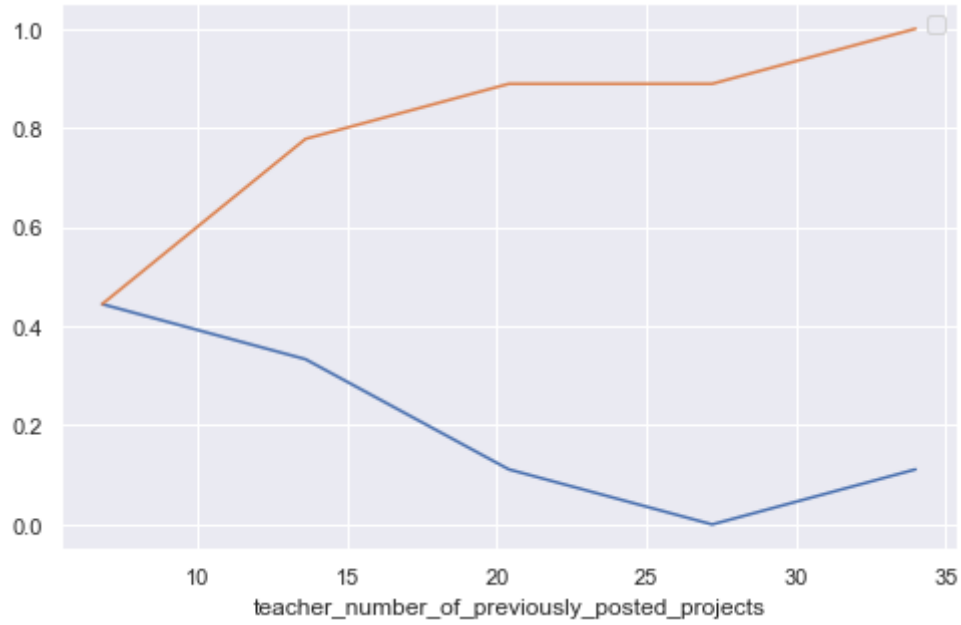
```
<matplotlib.axes._subplots.AxesSubplot at 0x2ae5e5a3308>
```



2.3.7 PDF & CDF with `teacher_number_of_previously_posted_projects` of fp

In [119]:

```
plt.figure(figsize=(8,5))
counts, bin_edges = np.histogram(X_test_fp['teacher_number_of_previously_posted_project
pdf = counts/sum(counts)
cdf=np.cumsum(pdf)
pdf_plot = plt.plot(bin_edges[1:],pdf)
cdf_plot = plt.plot(bin_edges[1:],cdf)
plt.legend([pdf_plot, cdf_plot], ["PDF", "CDF"])
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



2.4 Apply decisoon Tree on set4 `TFIDF W2V`

In [120]:

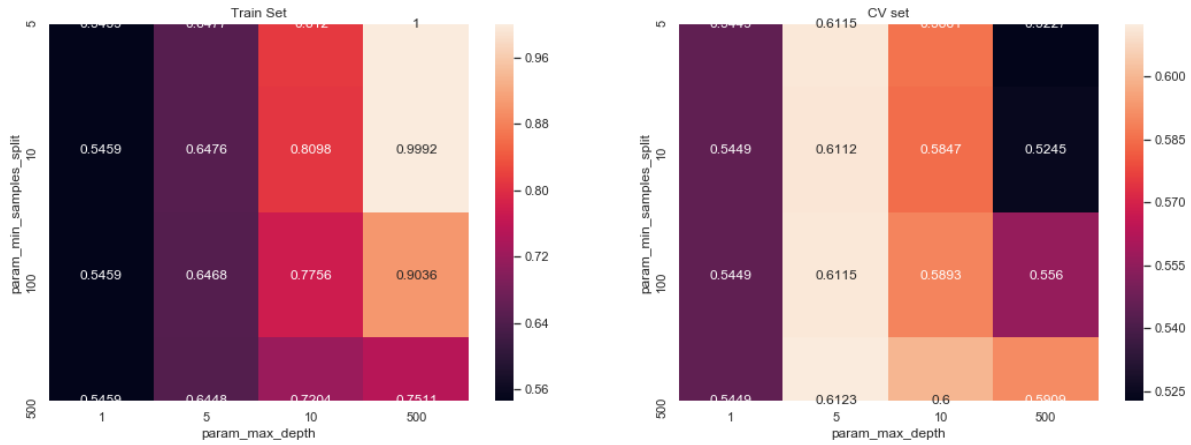
```
# Merging two sparse matrixs : https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matrix
# set_4 = encoded numarical + categorical + project_title_tfidf_w2v_vectors +essays_tfi

set4_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     preprocessed_essays_train_tfidf_w2v_vectors, preprocessed_project_
                     teacher_number_of_previously_posted_projects_normalizer_train,pric

set4_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, scl
                    teacher_prefix_one_hot_test, project_grade_category_one_hot_test,
                    preprocessed_project_title_test_tfidf_w2v_vectors,preprocessed_essay
                    teacher_number_of_previously_posted_projects_normalizer_test, price_

set4_cv = hstack((clean_categories_one_hot_cv,clean_subcategories_one_hot_cv, school_st
                  teacher_prefix_one_hot_cv, project_grade_category_one_hot_cv,
                  preprocessed_project_title_cv_tfidf_w2v_vectors,preprocessed_essays_cv
                  teacher_number_of_previously_posted_projects_normalizer_cv, price_norm

print("Final Data Matrix of set4 :")
print("shape of set4_train and y_train :", set4_train.shape , y_train.shape)
print("shape of set4_test and y_test   :", set4_test.shape , y_test.shape)
print("shape of set4_cv and y_cv       :", set4_cv.shape , y_cv.shape)
```

```
Final Data Matrix of set4 :
shape of set4_train and y_train : (49041, 701) (49041,)
shape of set4_test and y_test   : (36052, 701) (36052,)
shape of set4_cv and y_cv       : (24155, 701) (24155,)
```

2.4.1 Hyper parameter Tuning to find best estimator : GridSearchCV `set4 (TFIDF W2V)`

In [122]:

```
import seaborn as sns; sns.set()
max_score_4 = pd.DataFrame(clf4.cv_results_).groupby(['param_min_samples_split','param_
    'mean_test_score','mean_train_score']]

fig,ax = plt.subplots(1,2,figsize=(18,6))
sns.heatmap(max_score_4.mean_train_score, annot=True, fmt='.4g',ax=ax[0])
sns.heatmap(max_score_4.mean_test_score, annot=True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV set')
plt.show()
```



In [124]:

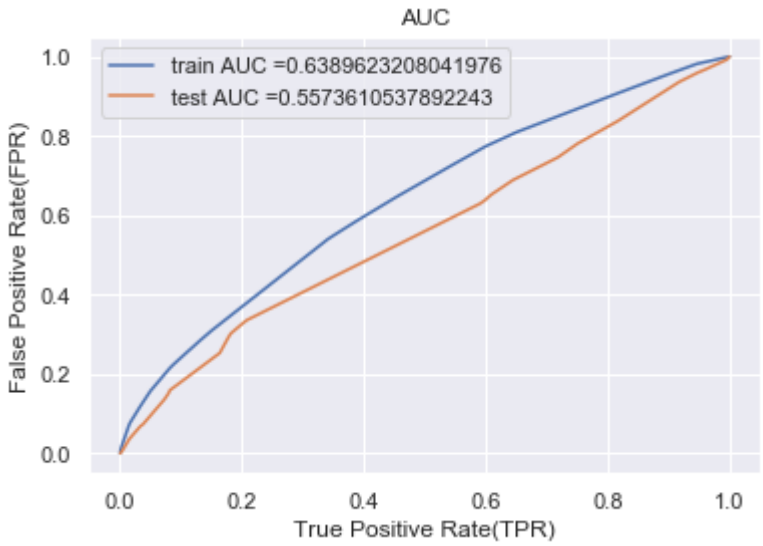
```
# best estimator and best parameters
print(clf4.best_estimator_)
print(clf4.score(set4_train,y_train))
print(clf4.score(set4_test, y_test))
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_dept
h=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')
0.6389623208041976
0.5573610537892243
```

2.4.2 Train model using the best hyper-parameter value `set4 TFIDF W2V`

In [125]:

```
%%time
from sklearn.metrics import auc,roc_curve

clf_4m =DecisionTreeClassifier (class_weight = 'balanced',max_depth=5,min_samples_split=
clf_4m.fit(set4_train,y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
# not the predicted outputs
train_fpr, train_tpr, thresholds = roc_curve(y_train,clf_4m.predict_proba(set4_train)[:
test_fpr, test_tpr, thresholds = roc_curve(y_test, clf_4m.predict_proba(set4_test)[:,1]

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.grid(True)
plt.xlabel("True Positive Rate(TPR)")
plt.ylabel("False Positive Rate(FPR)")
plt.title("AUC")
plt.show()
```



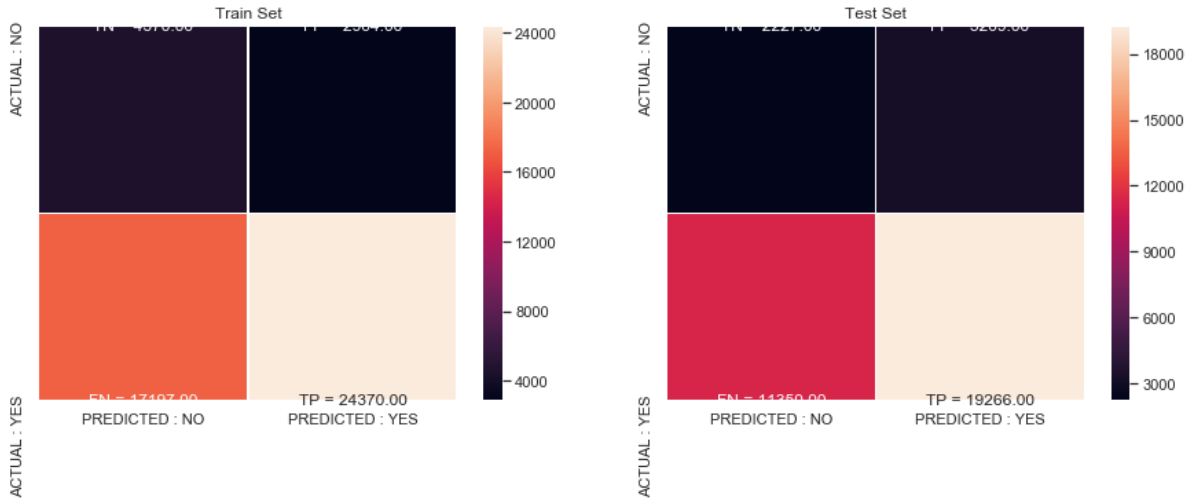

```
Wall time: 19.9 s
```

2.4.3 Confustion Matrix: set4_train and set4_test

In [126]:

```
clf_4m.fit(set4_train,y_train)
y_train_pred_4 = clf_4m.predict_proba(set4_train)[:,1]
y_test_pred_4 = clf_4m.predict_proba(set4_test)[:,1]
#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
conf_matr_df_train_4 = confusion_matrix(y_train, predict(y_train_pred_4, thresholds, tr
conf_matr_df_test_4 = confusion_matrix(y_test, predict(y_test_pred_4, thresholds, test_

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(k
labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(ke

sns.heatmap(conf_matr_df_train_4, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDIC
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''
sns.heatmap(conf_matr_df_test_4, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',
ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.3584841464733335 for threshold 0.491
the maximum value of tpr*(1-fpr) 0.26612484505064804 for threshold 0.547
```



2.4.4 Flase Possitive points

In [127]:

```
conf_matr_df_test_4
```

Out[127]:

```
array([[ 2227,  3209],
       [11350, 19266]], dtype=int64)
```

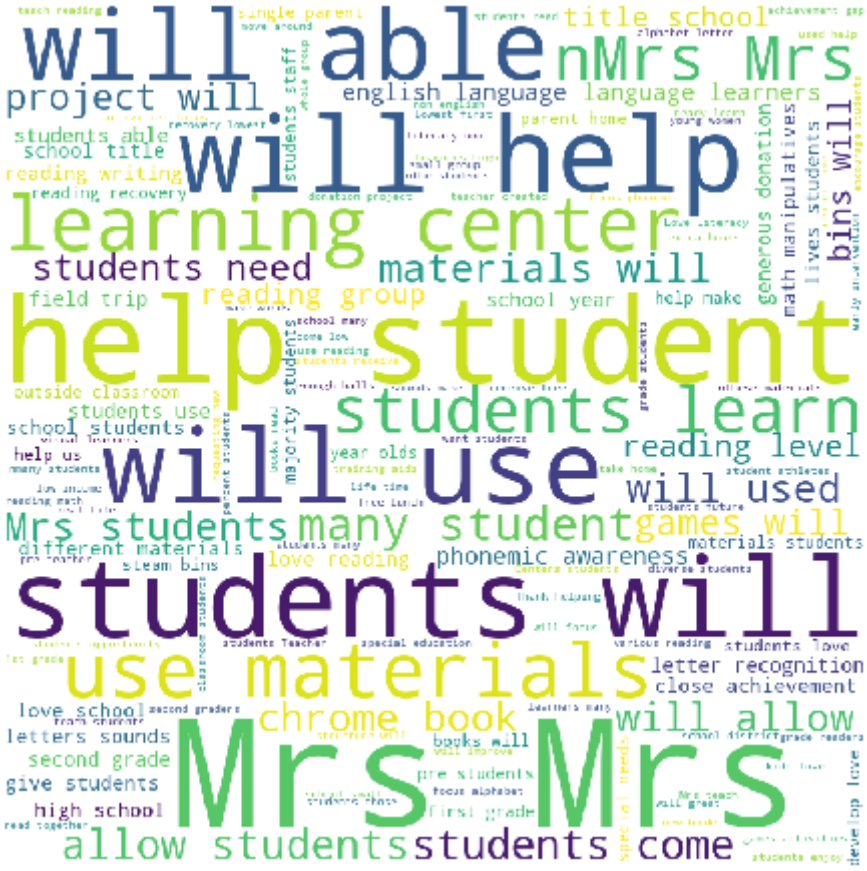
In [132]:

```
#Cite : https://datascience.stackexchange.com/questions/28493/confusion-matrix-get-item
fp =[]
for i in range(len(y_test)):
    if (y_test.values[i] == 0) & (y_test_pred_1[i] == 1) :
        fp.append(i)

fp_essay_4=[]
for i in fp:
    fp_essay_4.append(X_test['essay'].values[i])
```

2.4.5 Ploting the WordCloud with the words of essay text of these false positive data points

In [133]:

```
# Cite : https://www.geeksforgeeks.org/generating-word-cloud-python/
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay_4 : # iterate through the csv file
    val = str(val) # typecaste each val to string
    tokens = val.split() # split the value
    for i in range(len(tokens)): # Converts each token into lowercase
        tokens[i] = tokens[i].lower()
        for words in tokens :
            comment_words = comment_words + words + ' '
wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords =
# plot the WordCloud image
plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



2.4.6 Ploting Box plot with `price of false possitive points`

In [134]:

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

X_test_fp.head(1)
len(X_test_fp)
```

Out[134]:

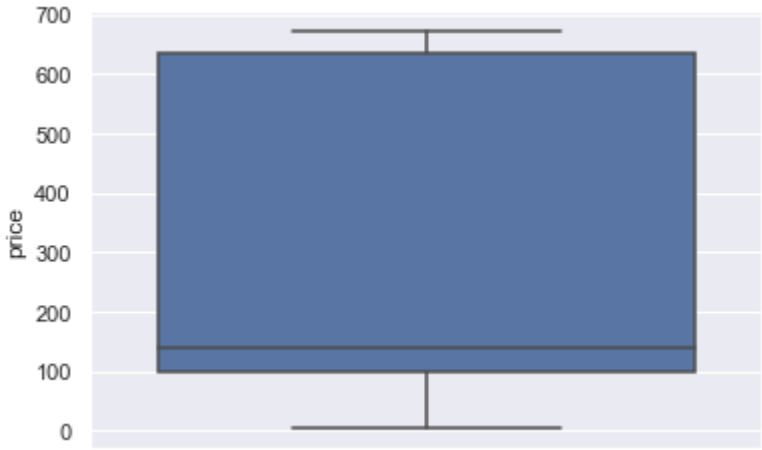
```
9
```

In [135]:

```
##Box Plot (FP 'price')
sns.boxplot(y='price', data=X_test_fp)
```

Out[135]:

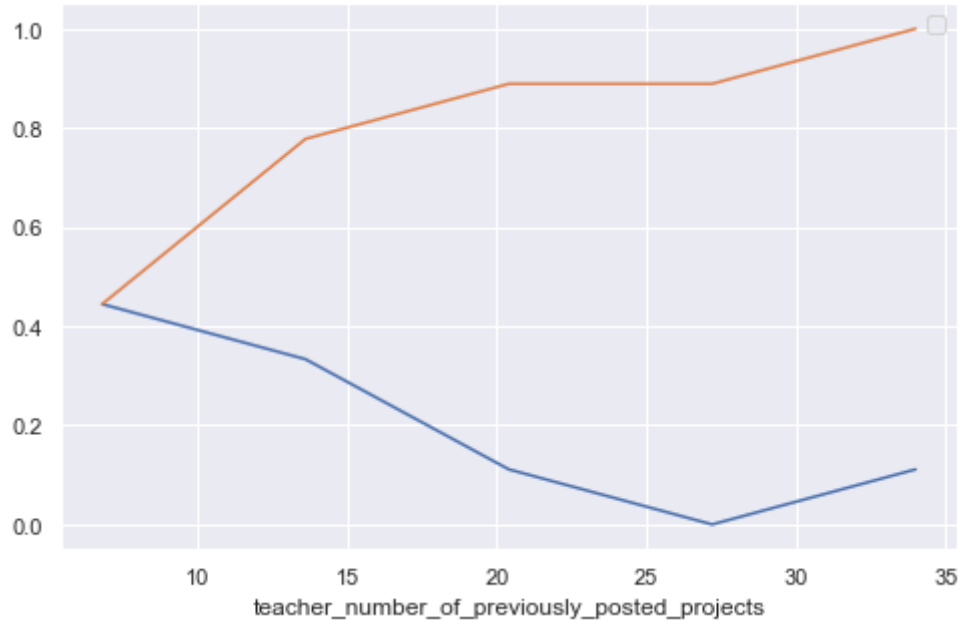
```
<matplotlib.axes._subplots.AxesSubplot at 0x2ae5ea87e08>
```



2.4.7 PDF & CDF with `teacher_number_of_previously_posted_projects` of fp

In [136]:

```
plt.figure(figsize=(8,5))
counts, bin_edges = np.histogram(X_test_fp['teacher_number_of_previously_posted_project
pdf = counts/sum(counts)
cdf=np.cumsum(pdf)
pdf_plot = plt.plot(bin_edges[1:],pdf)
cdf_plot = plt.plot(bin_edges[1:],cdf)
plt.legend([pdf_plot, cdf_plot], ["PDF", "CDF"])
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



2.5 [Task-2]Getting top 5k features using `feature_importances_with TFIDF(set2)`

Applying Decision tree on Important features

In [137]:

```
def selectKImportance(model, X, k=5):
    return X[:,model.best_estimator_.feature_importances_.argsort()[::-1][:k]]
```

In [138]:

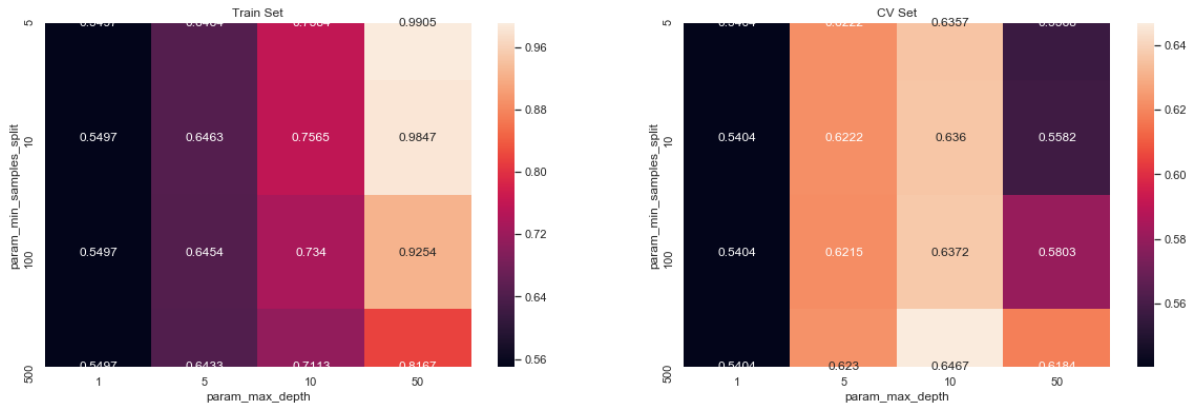
```
# for tf-idf set 2
set5_train = selectKImportance(clf2, set2_train,5000)
set5_test = selectKImportance(clf2, set2_test, 5000)
print(set5_train.shape)
print(set5_test.shape)
```

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

2.5.1 Hyper Parameter Tuning to find best estimator : GridSearchCV set5

In [139]:

```
dt5= DecisionTreeClassifier(class_weight = 'balanced')
parameters = {'max_depth': [1, 5, 10, 50], 'min_samples_split': [5, 10,100, 500]}
clf5 = GridSearchCV(dt5, parameters, cv=3, scoring='roc_auc',return_train_score=True)
set5_fit= clf5.fit(set5_train, y_train)
import seaborn as sns; sns.set()
max_scores5 = pd.DataFrame(clf5.cv_results_).groupby(['param_min_samples_split', 'param
    'mean_test_score', 'mean_train_score']]

fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores5.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores5.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [140]:

```
#Best Estimator and Best tune parameters
print(clf5.best_estimator_)
#Mean cross-validated score of the best_estimator
print(clf5.score(set5_train,y_train))
print(clf5.score(set5_test,y_test))
```

```
DecisionTreeClassifier(class_weight='balanced', criterion='gini', max_dept
h=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')
0.7065067707248756
0.6457383346234964
```

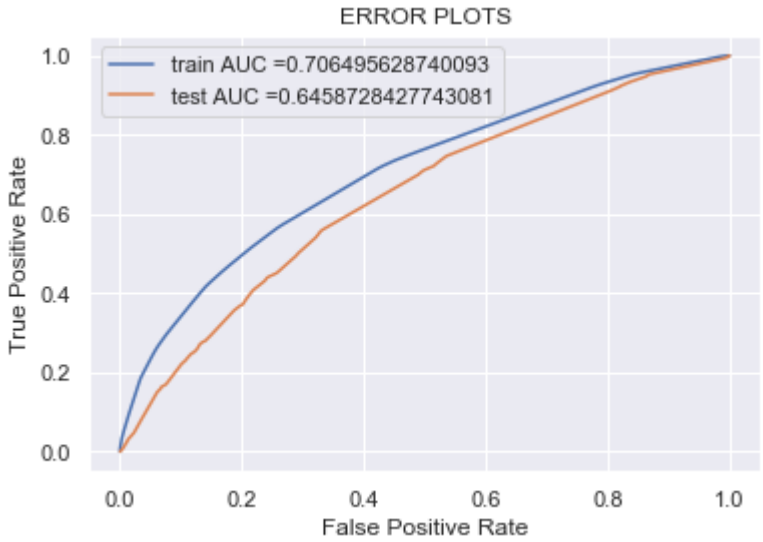
In [141]:

```
# Best tune parameters
best_tune_parameters=[{'max_depth': [10], 'min_samples_split':[500] } ]
```

2.5.2 Train model using best hyper parameter value set5

In [146]:

```
clf5= GridSearchCV( DecisionTreeClassifier(class_weight = 'balanced'),best_tune_paramet

clfV5=DecisionTreeClassifier (class_weight = 'balanced',max_depth=10,min_samples_split=

clf5.fit(set5_train, y_train)
# for visulation
clfV5.fit(set5_train, y_train)
#https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.ht
#sklearn.linear_model.SGDClassifier.decision_function
y_train_pred5 = clf5.predict_proba(set5_train) [:,1]
y_test_pred5 = clf5.predict_proba(set5_test) [:,1]

train_fpr, train_tpr, thresholds = roc_curve(y_train, y_train_pred5)
test_fpr, test_tpr, thresholds = roc_curve(y_test, y_test_pred5)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```

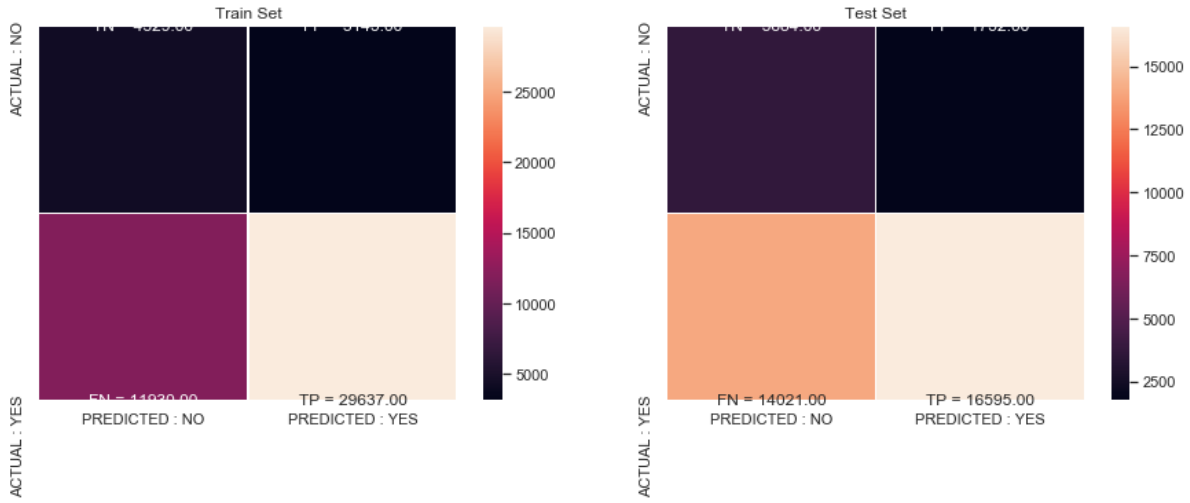


2.5.3 Confusion Matrix : Set5_train, set5_test

In [147]:

```
clf5.fit(set5_train,y_train)
y_train_pred_5 = clf5.predict_proba(set5_train)[:,1]
y_test_pred_5  = clf5.predict_proba(set5_test)[:,1]


#https://www.quantinsti.com/blog/creating-heatmap-using-python-seaborn
import seaborn as sns; sns.set()
conf_matr_df_train_5 = confusion_matrix(y_train, predict(y_train_pred_5, thresholds, tra
conf_matr_df_test_5  = confusion_matrix(y_test, predict(y_test_pred_5, thresholds, test_

key = (np.asarray([['TN','FP'], ['FN', 'TP']]))
fig, ax = plt.subplots(1,2, figsize=(15,5))

labels_train = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(ke

labels_test = (np.asarray(["{0} = {1:.2f}" .format(key, value) for key, value in zip(key

sns.heatmap(conf_matr_df_train_5, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICT
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_train, fmt = ''

sns.heatmap(conf_matr_df_test_5, linewidths=.5, xticklabels=['PREDICTED : NO', 'PREDICTE
            yticklabels=['ACTUAL : NO', 'ACTUAL : YES'], annot = labels_test, fmt = '',

ax[0].set_title('Train Set')
ax[1].set_title('Test Set')
plt.show()
```

```
the maximum value of tpr*(1-fpr) 0.4204745382745734 for threshold 0.475
the maximum value of tpr*(1-fpr) 0.3740586832876585 for threshold 0.537
```



2.5.4 False Possitive points

In [148]:

```
conf_matr_df_test_5
```

Out[148]:

```
array([[ 3684,  1752],
       [14021, 16595]], dtype=int64)
```

In [151]:

```
#Cite : https://datascience.stackexchange.com/questions/28493/confusion-matrix-get-item
fp =[]
for i in range(len(y_test)):
    if (y_test.values[i] == 0) & (y_test_pred_5[i] == 1) :
        fp.append(i)
fp_essay_5=[]
for i in fp:
    fp_essay_5.append(X_test['essay'].values[i])
print(fp)
```

```
[1711, 9479, 10591, 12372, 14245, 14254, 14336, 17873, 19502, 22344, 2286
1, 26218, 26337, 26515, 29868, 30162, 30870, 34778]
```

2.5.5 Ploting the WordCloud with the words of essay text of these false positive data points

In [152]:

```
# Cite : https://www.geeksforgeeks.org/generating-word-cloud-python/
from wordcloud import WordCloud, STOPWORDS
comment_words = ' '
stopwords = set(STOPWORDS)
for val in fp_essay_5 : # iterate through the csv file
    val = str(val) # typecaste each val to string
    tokens = val.split() # split the value
    for i in range(len(tokens)): # Converts each token into lowercase
        tokens[i] = tokens[i].lower()
        for words in tokens :
            comment_words = comment_words + words + ' '
wordcloud = WordCloud(width = 800, height = 800, background_color ='white', stopwords =
# plot the WordCloud image
plt.figure(figsize = (6, 6), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```

In [153]:

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

X_test_fp.head(1)
```

Out[153]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	
9479	175709	p104623	fa49e9681d1419dcb4a00c7aa49cff15	Mrs.	IL	2 0 11:5

2.5.6 Ploting Box-plot with price of false possitive points

In [154]:

```
##Box Plot (FP 'price')
sns.boxplot(y='price', data=X_test_fp)
```

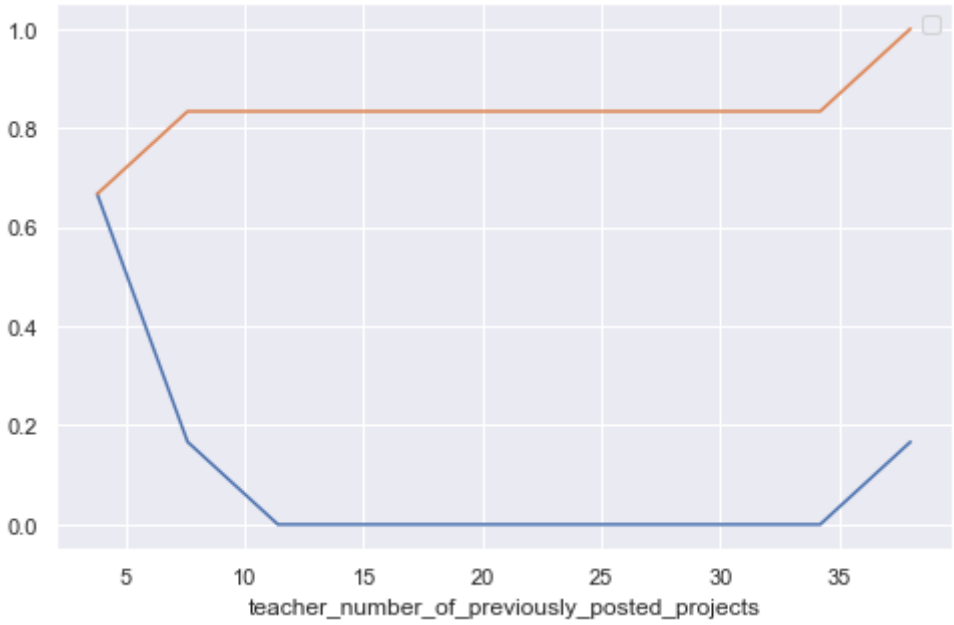
Out[154]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2ae5efa1608>
```

2.5.7 PDF & CDF with teacher_number_of_previously_posted_projects of fp

In [155]:

```
plt.figure(figsize=(8,5))
counts, bin_edges = np.histogram(X_test_fp['teacher_number_of_previously_posted_project
pdf = counts/sum(counts)
cdf=np.cumsum(pdf)
pdf_plot = plt.plot(bin_edges[1:],pdf)
cdf_plot = plt.plot(bin_edges[1:],cdf)
plt.legend([pdf_plot, cdf_plot], ["PDF", "CDF"])
plt.xlabel('teacher_number_of_previously_posted_projects')
plt.show()
```



3.A .Observation:

1. By using important data points (5000) from set2_tfidf, we got the same performence of 64%.

3.B. Conclusion:

In [1]:

```
# Link : http://zetcode.com/python/prettytable/
from prettytable import PrettyTable
p = PrettyTable()

p.field_names = ["Vectorizer", "Model", "Best_param_max_depth","Best_param_min_samples_

p.add_row(["BOW-Experiment-set1","DT",10, 500, 0.70, 0.64])
p.add_row(["TFIDF-set2","DT",10, 500, 0.70, 0.64])
p.add_row(["AVG W2V-Experiment set3","DT",5, 500, 0.64, 0.61])
p.add_row(["TFIDF W2V - set4","DT",5, 500, 0.63, 0.55])
p.add_row(["Important 5K points TFIDF(set2) - Set5","DT",10, 500, 0.70, 0.64])

print(p)
```

```
+----------------------------------------+-------+----------------------+-
----------------------------+-----------+----------+
|               Vectorizer               | Model | Best_param_max_depth |
Best_param_min_samples_split | Train AUC | Test AUC |
+----------------------------------------+-------+----------------------+-
----------------------------+-----------+----------+
|          BOW-Experiment-set1           |   DT  |          10          |
500                          |    0.7    |   0.64   |
|               TFIDF-set2               |   DT  |          10          |
500                          |    0.7    |   0.64   |
|        AVG W2V-Experiment set3         |   DT  |          5           |
500                          |    0.64   |   0.61   |
|            TFIDF W2V - set4            |   DT  |          5           |
500                          |    0.63   |   0.55   |
| Important 5K points TFIDF(set2) - Set5 |   DT  |          10          |
500                          |    0.7    |   0.64   |
+----------------------------------------+-------+----------------------+-
----------------------------+-----------+----------+
```

Thank You.

Sign Off RAMESH BATTU (https://www.linkedin.com/in/rameshbattuai/)