

# 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	
<code>project_id</code>	A unique identifier for the proposed project
<code>project_title</code>	Title of the project
<code>project_grade_category</code>	Grade level of students for which the project is targeted
<code>teacher_name</code>	Name of the teacher who submitted the project
<code>school_name</code>	Name of the school where the project is located
<code>project_submitted_date</code>	Date the project was submitted
<code>project_state</code>	State where the project is located
<code>project_is_approved</code>	Whether the project was approved or not

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
<b>id</b>	A project_id value from the train.csv file. <b>Example:</b> p036502
<b>description</b>	Description of the resource. <b>Example:</b> Tenor Saxophone Reeds, Box of 25
<b>quantity</b>	Quantity of the resource required. <b>Example:</b> 3
<b>price</b>	Price of the resource required. <b>Example:</b> 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
<code>project_is_approved</code>	A binary flag indicating whether DonorsChoose approved the project. A value of <code>0</code> indicates the project was not approved, and a value of <code>1</code> indicates the project was approved.

## Notes on the Essay Data

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

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

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

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

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

```
In [1]: 1 import os
2 os.chdir("F:\f\MY_____AAIC\AssignmentS\DonorsChoose_Data")
3 os.getcwd()
```

Out[1]: 'F:\\f\\MY\_\_\_\_\_\\AAIC\\AssignmentS\\DonorsChoose\_Data'

```
In [2]: 1 %matplotlib inline
2 import warnings
3 warnings.filterwarnings("ignore")
4
5 import sqlite3
6 import pandas as pd
7 import numpy as np
8 import nltk
9 import string
10 import matplotlib.pyplot as plt
11 import seaborn as sns
12 from sklearn.feature_extraction.text import TfidfTransformer
13 from sklearn.feature_extraction.text import TfidfVectorizer
14
15 from sklearn.feature_extraction.text import CountVectorizer
16 from sklearn.metrics import confusion_matrix
17 from sklearn import metrics
18 from sklearn.metrics import roc_curve, auc
19 from nltk.stem.porter import PorterStemmer
20
21 import re
22 # Tutorial about Python regular expressions: https://pymotw.com/2/re/
23 import string
24 from nltk.corpus import stopwords
25 from nltk.stem import PorterStemmer
26 from nltk.stem.wordnet import WordNetLemmatizer
27
28 from gensim.models import Word2Vec
29 from gensim.models import KeyedVectors
30 import pickle
31
32 from tqdm import tqdm
33 import os
34
35 from plotly import plotly
36 import plotly.offline as offline
37 import plotly.graph_objs as go
38 offline.init_notebook_mode()
39 from collections import Counter
```

C:\Users\Anvesh\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning:  
detected Windows; aliasing chunkize to chunkize\_serial  
warnings.warn("detected Windows; aliasing chunkize to chunkize\_serial")

## 1.1 Reading Data

```
In [3]: 1 project_data = pd.read_csv('train_data.csv')
2 resource_data = pd.read_csv('resources.csv')
```

```
In [4]: 1 print("Number of data points in train data", project_data.shape)
2 print('-'*50)
3 print("The attributes of data :", project_data.columns.values)
```

Number of data points in train data (109248, 17)  
-----  
The attributes of data : ['Unnamed: 0' 'id' 'teacher\_id' 'teacher\_prefix' 'school\_state'  
'project\_submitted\_datetime' 'project\_grade\_category'  
'project\_subject\_categories' 'project\_subject\_subcategories'  
'project\_title' 'project\_essay\_1' 'project\_essay\_2' 'project\_essay\_3'  
'project\_essay\_4' 'project\_resource\_summary'  
'teacher\_number\_of\_previously\_posted\_projects' 'project\_is\_approved']

```
In [5]: 1 print("Number of data points in train data", resource_data.shape)
2 print(resource_data.columns.values)
3 resource_data.head(2)
```

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

Out[5]:

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

## 1.2 preprocessing of project\_subject\_categories

```
In [6]:  
1 catogories = list(project_data['project_subject_categories'].values)  
2 # remove special characters from list of strings python: https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-list-of-strings-in-python  
3  
4 # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/  
5 # https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-list-of-strings-in-python  
6 # https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string  
7 cat_list = []  
8 for i in catogories:  
9     temp = ""  
10    # consider we have text Like this "Math & Science, Warmth, Care & Hunger"  
11    for j in i.split(','):# it will split it in three parts ["Math & Scienc  
12        if 'The' in j.split(): # this will split each of the category based  
13            j=j.replace('The','') # if we have the words "The" we are going  
14            j = j.replace(' ','') # we are placeing all the ' '(space) with ''(e  
15            temp+=j.strip()+" "# abc ".strip() will return "abc", remove the t  
16            temp = temp.replace('&','_') # we are replacing the & value into  
17        cat_list.append(temp.strip())  
18  
19 project_data['clean_categories'] = cat_list  
20 project_data.drop(['project_subject_categories'], axis=1, inplace=True)  
21  
22 from collections import Counter  
23 my_counter = Counter()  
24 for word in project_data['clean_categories'].values:  
25     my_counter.update(word.split())  
26  
27 cat_dict = dict(my_counter)  
28 sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))  
29
```

## 1.3 preprocessing of project\_subject\_subcategories

```
In [7]: 1 sub_catogories = list(project_data['project_subject_subcategories'].values)
2 # remove special characters from list of strings python: https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-list-in-python
3
4 # https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
5 # https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-list-in-python
6 # https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string
7
8 sub_cat_list = []
9 for i in sub_catogories:
10     temp = ""
11     # consider we have text Like this "Math & Science, Warmth, Care & Hunger"
12     for j in i.split(','):# it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
13         if 'The' in j.split():# this will split each of the category based on 'The'
14             j=j.replace('The','') # if we have the words "The" we are going to remove it
15             j = j.replace(' ','') # we are placeing all the ' ' (space) with ''(empty string)
16             temp +=j.strip()+" "# abc ".strip() will return "abc", remove the trailing space
17             temp = temp.replace('&','_')
18     sub_cat_list.append(temp.strip())
19
20 project_data['clean_subcategories'] = sub_cat_list
21 project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
22
23 # count of all the words in corpus python: https://stackoverflow.com/a/228984
24 my_counter = Counter()
25 for word in project_data['clean_subcategories'].values:
26     my_counter.update(word.split())
27
28 sub_cat_dict = dict(my_counter)
29 sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

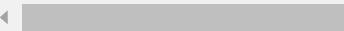
## 1.3 Text preprocessing

```
In [8]: 1 # merge two column text dataframe:
2 project_data["essay"] = project_data["project_essay_1"].map(str) + \
3                         project_data["project_essay_2"].map(str) + \
4                         project_data["project_essay_3"].map(str) + \
5                         project_data["project_essay_4"].map(str)
```

```
In [9]: 1 project_data.head(2)
```

Out[9]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221 p253737 c90749f5d961ff158d4b4d1e7dc665fc		Mrs.	IN	20
1	140945 p258326 897464ce9ddc600bcfd1151f324dd63a		Mr.	FL	20



```
In [10]: 1 ##### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [11]:

```

1 # printing some random reviews
2 print(project_data['essay'].values[0])
3 print("*50")
4 print(project_data['essay'].values[150])
5 print("*50")
6 print(project_data['essay'].values[1000])
7 print("*50")
8 print(project_data['essay'].values[20000])
9 print("*50")
10 print(project_data['essay'].values[99999])
11 print("*50")

```

My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 languages represented in our English Learner program with students at every level of mastery. We also have over 40 countries represented with the families within our school. Each student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, beliefs, and respect.\\"The limits of your language are the limits of your world.\\"-Ludwig Wittgenstein Our English learners have a strong support system at home that begs for more resources. Many times our parents are learning to read and speak English along side of their children. Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English language even if no one at home is able to assist. All families with students within the Level 1 proficiency status, will be offered to be a part of this program. These educational videos will be specially chosen by the English Learner Teacher and will be sent home regularly to watch. The videos are to help the child develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and educational dvd's for the years to come for other EL students.\r\n\r\n=====

The 51 fifth grade students that will cycle through my classroom this year all love learning, at least most of the time. At our school, 97.3% of the students receive free or reduced price lunch. Of the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves to get together and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by the students, dances, and games. At the end of the year the school hosts a carnival to celebrate the hard work put in during the school year, with a dunk tank being the most popular activity. My students will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged chairs. As I will only have a total of ten in the classroom and not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as special chairs students will each use on occasion. I will utilize them in place of chairs at my small group tables during math and reading times. The rest of the day they will be used by the students who need the highest amount of movement in their life in order to stay focused on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki Stools. They can't get their fill of the 5 stools we already have. When the students are sitting in group with me on the Hokki Stools, they are always moving, but at the same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be taken. There are always students who head over to the kidney table to get one of the stools who are disappointed as there are not enough of them. \r\n\r\n

e ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the same time. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit still.nannan

---

How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an "open classroom" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more. With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan

---

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

---

The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school has 803 students which is makeup is 97.6% African-American, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2% African-American students. Most of the students are on free or reduced lunch. We are n't receiving doctors, lawyers, or engineers children from rich backgrounds or

neighborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smart, effective, efficient, and disciplined students with good character. In our classroom we can utilize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the volume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause and replay it at any time.\r\nThe cart will allow me to have more room for storage of things that are needed for the day and has an extra part to it I can use. The table top chart has all of the letter, words and pictures for students to learn about different letters and it is more accessible.nannan

---

## 1.4 Train\_Test\_Split

In [12]:

```

1 #Train Test Split
2 from sklearn.model_selection import train_test_split
3 x_train, x_test, y_train, y_test = train_test_split(project_data, project_da-
4                                     test_size = 0.33, strati-
5                                     random_state = 42)

```

In [13]:

```

1 #Train CV Split
2 x_train, x_cv, y_train, y_cv = train_test_split(x_train, y_train, test_size-
3                                                 random_state = 42)

```

In [14]:

```

1 print(x_test.columns)
2 print(x_train.columns)
3 #print(x_cv.columns)
4 #print(x_train.shape)
5 #print(x_test.shape)
6 #print(x_cv.shape)

```

```

Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', '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', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', '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', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')

```

In [15]:

```

1 #Dropping Class Label in train test and cv data
2 x_train.drop(["project_is_approved"], axis = 1, inplace = True)
3 x_test.drop(["project_is_approved"], axis = 1, inplace = True)
4 x_cv.drop(["project_is_approved"], axis = 1, inplace = True)

```

```
In [16]: 1 print(x_train.columns)
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', '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'],
      dtype='object')
```

```
In [ ]: 1
```

## Preparing Data for Models

```
In [17]: 1 # https://stackoverflow.com/a/47091490/4084039
2 import re
3
4 def decontracted(phrase):
5     # specific
6     phrase = re.sub(r"won't", "will not", phrase)
7     phrase = re.sub(r"can't", "can not", phrase)
8
9     # general
10    phrase = re.sub(r"\n\t", " not", phrase)
11    phrase = re.sub(r"\'re", " are", phrase)
12    phrase = re.sub(r"\'s", " is", phrase)
13    phrase = re.sub(r"\d", " would", phrase)
14    phrase = re.sub(r"\ll", " will", phrase)
15    phrase = re.sub(r"\t", " not", phrase)
16    phrase = re.sub(r"\'ve", " have", phrase)
17    phrase = re.sub(r"\m", " am", phrase)
18
19    return phrase
```

```
In [18]: 1 sent = decontracted(project_data['essay'].values[20000])
2 print(sent)
3 print("=*50)
```

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

---

```
In [19]: 1 # \r \n \t remove from string python: http://texthandler.com/info/remove-line
2 sent = sent.replace('\\r', ' ')
3 sent = sent.replace('\\t', ' ')
4 sent = sent.replace('\\n', ' ')
5 print(sent)
```

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

In [20]:

```

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

```

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

In [21]:

```

1 # https://gist.github.com/sebleier/554280
2 # we are removing the words from the stop words list: 'no', 'nor', 'not'
3 stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'yo
4 "you'll", "you'd", "your", "yours", "yourself", "yourselves", 'h
5 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', '
6 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
7 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
8 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'bec
9 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into'
10 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
11 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',
12 'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 't
13 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shou
14 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn'
15 "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
16 "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "sho
17 'won', "won't", 'wouldn', "wouldn't"]

```

## Project\_Essays preprocessing

In [22]:

```

1 #train_preprocessed_essays
2 # Combining all the above students
3 from tqdm import tqdm
4 train_preprocessed_essays = []
5 # tqdm is for printing the status bar
6 for sentance in tqdm(x_train['essay'].values):
7     sent = decontracted(sentance)
8     sent = sent.replace('\r', ' ')
9     sent = sent.replace('\\"', ' ')
10    sent = sent.replace('\\n', ' ')
11    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
12    # https://gist.github.com/sebleier/554280
13    sent = ' '.join(e for e in sent.split() if e not in stopwords)
14    train_preprocessed_essays.append(sent.lower().strip())

```

100% | 49041/49041 [00:38<00:00, 1259.34it/s]

In [23]:

```
1 train_preprocessed_essays[10]
```

Out[23]: 'keeping visual arts classrooms passion i want share classroom in room add means appliance door decor providing quality supplies foster appreciation desire pursue art career i work title one school suburban neighborhood we provide many magnet programs including pbi positive behavior intervention preschool el err i worked many students potential become artists i would like provide experiences quality materials uniform colors brushes not shed bristles chance paint real canvas i hardly wait share opportunity paint room 30 students not opportunity mix colors create real canvas we start outdoor landscapes sky water reflection silhouette trees begin three colors red white black mix need provide contrast high lights 3 colors we cover vocabulary words tints shades hue line shape foreground mid ground background practice plain construction paper create final product canvas we begin implementation performance based art report card next year to provide instruction color mixing techniques students could realize final project real canvas would positive impact students may grow become design engineers architects fashion designers tomorrow'

In [24]:

```

1 # test_preprocessed_essay
2 from tqdm import tqdm
3 test_preprocessed_essays = []
4 # tqdm is for printing the status bar
5 for sentance in tqdm(x_test['essay'].values):
6     sent = decontracted(sentance)
7     sent = sent.replace('\r', ' ')
8     sent = sent.replace('\\"', ' ')
9     sent = sent.replace('\\n', ' ')
10    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
11    # https://gist.github.com/sebleier/554280
12    sent = ' '.join(e for e in sent.split() if e not in stopwords)
13    test_preprocessed_essays.append(sent.lower().strip())

```

100% | 36052/36052 [00:29<00:00, 1217.63it/s]

In [25]: 1 test\_preprocessed\_essays[10]

Out[25]: 'hey i astronaut look says one kindergarten students no look i engineer says an other another 5 year old shouts i love technology want computer person i grow t his life eyes 5 6 year old public school boys girls they love learning explorin g pretending want grow the children come variety ethnic social cultural economi c backgrounds but children one thing common love learning especially math scien ce our motto grown anything want learning anything possible just witnessing des ire learn math science rewarding my students curious world need know technology help identify real world become scientist math teacher engineer my students lov e hands activities motivated technology they love learning new things enthusias m radiates class this ipad case make huge difference way students learning math science it let customize learning according students levels level level level i t allow children opportunity understand real world live the children use free s cience app brain pop jr see scientist work lab astronauts go outer space develo p computer programs help keep earth free pollution they use free math app sushi monster strengthen reasoning skills learn strategies solving math problems unde rstanding patterns world this ipad allow show students stem activites used diff erentiate instruction students bring real world classroom they learn work addit ion math problems 5 3 2 10 seeing robot build tower learn planet mars water see ing pictures form hubble telescope learn ipad show patterns exist throughout wo rld this ipad used kindergarten math science exploration center the children pi ck ipad go free stem app like brain pop jr using fingers activate game stimulat e enrich learning math science nannan'

In [26]: 1 # CV\_preprocessed\_essays

```

1 # CV_preprocessed_essays
2 from tqdm import tqdm
3 cv_preprocessed_essays = []
4 # tqdm is for printing the status bar
5 for sentance in tqdm(x_cv['essay'].values):
6     sent = decontracted(sentance)
7     sent = sent.replace('\r', ' ')
8     sent = sent.replace('\n', ' ')
9     sent = sent.replace('\n', ' ')
10    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
11    # https://gist.github.com/sebleier/554280
12    sent = ' '.join(e for e in sent.split() if e not in stopwords)
13    cv_preprocessed_essays.append(sent.lower().strip())

```

100% | 24155/24155 [00:19<00:00, 1231.15it/s]

In [27]: 1 cv\_preprocessed\_essays[10]

Out[27]: 'when asked music means student told music makes feel happy it changed i not anything express feelings i this hope every student joins class my biggest challenge providing everything need learn i blessed able work wonderful students they hunger music insatiable each day i able go work inspired dedication hard work students put everything our school made mission build culture embraces musical learning give every student opportunity participate music our school low income area best provide much needed supplies possible music program keeps growing struggling keep there nothing worse hearing violins played tune it difficult beginning violin students tune strings not always hear differences sound this means teacher often ends tuning instruments taking valuable learning time if students pitch pipes would able play individual notes match violin strings sounds this means could also fix violins get tune practicing home resulting better practice improvement individual playing nannan'

## Project\_Titles\_preprocessing

In [28]:

```

1 # train_preprocessed_title
2 from tqdm import tqdm
3 train_preprocessed_titles = []
4 # tqdm is for printing the status bar
5 for sentance in tqdm(x_train['project_title'].values):
6     sent = decontracted(sentance)
7     sent = sent.replace('\r', ' ')
8     sent = sent.replace('\n', ' ')
9     sent = sent.replace('\n', ' ')
10    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
11    # https://gist.github.com/sebleier/554280
12    sent = ' '.join(e for e in sent.split() if e not in stopwords)
13    train_preprocessed_titles.append(sent.lower().strip())

```

100% |  
49041/49041 [00:01<00:00, 27203.95it/s]

In [29]: 1 train\_preprocessed\_titles[10]

Out[29]: 'we all have rembrandt inside us'

In [30]:

```

1 # Test_preprocessed_essays
2 from tqdm import tqdm
3 test_preprocessed_titles = []
4 # tqdm is for printing the status bar
5 for sentance in tqdm(x_test['project_title'].values):
6     sent = decontracted(sentance)
7     sent = sent.replace('\r', ' ')
8     sent = sent.replace('\n', ' ')
9     sent = sent.replace('\n', ' ')
10    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
11    # https://gist.github.com/sebleier/554280
12    sent = ' '.join(e for e in sent.split() if e not in stopwords)
13    test_preprocessed_titles.append(sent.lower().strip())

```

100% |

36052/36052 [00:01&lt;00:00, 27482.96it/s]

In [31]:

1 test\_preprocessed\_titles[10]

Out[31]:

'ipads ipads exploring'

In [32]:

```

1 # CV_preprocessed_titles
2 from tqdm import tqdm
3 cv_preprocessed_titles = []
4 # tqdm is for printing the status bar
5 for sentance in tqdm(x_cv['project_title'].values):
6     sent = decontracted(sentance)
7     sent = sent.replace('\r', ' ')
8     sent = sent.replace('\n', ' ')
9     sent = sent.replace('\n', ' ')
10    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
11    # https://gist.github.com/sebleier/554280
12    sent = ' '.join(e for e in sent.split() if e not in stopwords)
13    cv_preprocessed_titles.append(sent.lower().strip())

```

100% |

24155/24155 [00:00&lt;00:00, 27205.89it/s]

In [33]:

1 # after preprocesing
2 cv\_preprocessed\_titles[10]

Out[33]:

'help us play our violins in tune'

In [34]:

1 project\_data.columns

Out[34]:

```

Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', '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', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      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 (optional)
  
- quantity : numerical (optional)
- teacher\_number\_of\_previously\_posted\_projects : numerical
- price : numerical

### 1.5.1 Vectorizing Categorical data

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

In [101]:

```

1 # we use count vectorizer to convert the values into one
2 # Vectorizing Clean Categories
3 from sklearn.feature_extraction.text import CountVectorizer
4 vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=True)
5
6 vectorizer.fit(x_train['clean_categories'].values)
7 train_categories_one_hot = vectorizer.transform(x_train['clean_categories'])
8 test_categories_one_hot = vectorizer.transform(x_test['clean_categories'].values)
9 cv_categories_one_hot = vectorizer.transform(x_cv['clean_categories'].values)
10
11 print(vectorizer.get_feature_names())
12 print("Shape of Train matrix after one hot encoding ",train_categories_one_hot.shape)
13 print("Shape of Test matrix after one hot encoding ",test_categories_one_hot.shape)
14 print("Shape of cv matrix after one hot encoding ",cv_categories_one_hot.shape)

['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of Train matrix after one hot encoding (49041, 9)
Shape of Test matrix after one hot encoding (36052, 9)
Shape of cv matrix after one hot encoding (24155, 9)

```

In [102]:

```

1 # we use count vectorizer to convert the values into one
2 vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=True)
3 vectorizer.fit(x_train["clean_subcategories"].values)
4 train_sub_categories_one_hot = vectorizer.transform(x_train['clean_subcategories'])
5 test_sub_categories_one_hot = vectorizer.transform(x_test['clean_subcategories'])
6 cv_sub_categories_one_hot = vectorizer.transform(x_cv['clean_subcategories'])
7
8 print(vectorizer.get_feature_names())
9 print("Shape of Train matrix after Trainone hot encodig ",train_sub_categories_one_hot.shape)
10 print("Shape of test matrix after one hot encodig ",test_sub_categories_one_hot.shape)
11 print("Shape of cv_ matrix after one hot encodig ",cv_sub_categories_one_hot.shape)

```

```

['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of Train matrix after Trainone hot encodig (49041, 30)
Shape of test matrix after one hot encodig (36052, 30)
Shape of cv_ matrix after one hot encodig (24155, 30)

```

In [103]:

```

1 # you can do the similar thing with state, teacher_prefix and project_grade_
2 from collections import Counter
3 my_counter = Counter()
4 for word in project_data["school_state"].values:
5     my_counter.update(word.split())

```

In [104]:

```

1 # dict sort by value python: https://stackoverflow.com/a/613218/4084039
2 state_cat_dict = dict(my_counter)
3 sorted_state_cat_dict = dict(sorted(state_cat_dict.items(), key=lambda kv:

```

In [105]:

```

1 # Please do the similar feature encoding with state, teacher_prefix and project_grade_
2 #Using Count Vectorizer to convert the state value onto on hot encoded feature
3 vectorizer = CountVectorizer(vocabulary=list(sorted_state_cat_dict.keys()), lowercase=True)
4 vectorizer.fit(project_data['school_state'].values)
5 print(vectorizer.get_feature_names())
6
7
8 train_state_one_hot = vectorizer.transform(x_train['school_state'].values)
9 test_state_one_hot = vectorizer.transform(x_test['school_state'].values)
10 cv_state_one_hot = vectorizer.transform(x_cv['school_state'].values)
11
12 print("Shape of Train matrix after one hot encodig ",train_state_one_hot.shape)
13 print("Shape of Test matrix after one hot encodig ",test_state_one_hot.shape)
14 print("Shape of CV matrix after one hot encodig ",cv_state_one_hot.shape)

```

```

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

```

```
In [106]: 1 #https://stackoverflow.com/questions/42224700/attributeerror-float-object-ha
2 project_data['project_grade_category']=project_data['project_grade_category']
3 my_counter = Counter()
4 for word in project_data['project_grade_category'].values:
5     my_counter.update(word.split())
```

```
In [107]: 1 project_cat_dict = dict(my_counter)
2 sorted_project_cat_dict = dict(sorted(project_cat_dict.items(), key=lambda k
```

```
In [108]: 1 # feature encoding for project_grade_category also
2 vectorizer = CountVectorizer(vocabulary=list(sorted_project_cat_dict.keys()))
3 vectorizer.fit(project_data['project_grade_category'].values)
4 print(vectorizer.get_feature_names())
5
6 train_grade_one_hot = vectorizer.transform(x_train['project_grade_category'])
7 test_grade_one_hot = vectorizer.transform(x_test['project_grade_category'])
8 cv_grade_one_hot = vectorizer.transform(x_cv['project_grade_category'].value
9
10 print("Shape of Train matrix after one hot encodig ",train_grade_one_hot.sha
11 print("Shape of test matrix after one hot encodig ",test_grade_one_hot.shape
12 print("Shape of cv matrix after one hot encodig ",cv_grade_one_hot.shape)
```

['9-12', '6-8', '3-5', 'PreK-2', 'Grades']  
 Shape of Train matrix after one hot encodig (49041, 5)  
 Shape of test matrix after one hot encodig (36052, 5)  
 Shape of cv matrix after one hot encodig (24155, 5)

```
In [109]: 1 #https://stackoverflow.com/questions/42224700/attributeerror-float-object-ha
2 project_data['teacher_prefix']=project_data['teacher_prefix'].fillna(" ")
```

```
In [110]: 1 my_counter = Counter()
2 for word in project_data['teacher_prefix'].values:
3     my_counter.update(word.split())
```

```
In [111]: 1 # dict sort by value python: https://stackoverflow.com/a/613218/4084039
2 teacher_cat_dict = dict(my_counter)
3 sorted_teacher_cat_dict = dict(sorted(teacher_cat_dict.items(), key=lambda k
```

In [112]:

```

1 #Using Count Vectorizer to convert the teacher_prefix value onto on hot encoding
2 #ValueError: np.nan is an invalid document, expected byte or unicode string.
3 #https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learn
4 vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_cat_dict.keys()))
5 vectorizer.fit(project_data['teacher_prefix'].values.astype('U'))
6 print(vectorizer.get_feature_names())
7
8 train_teacher_one_hot = vectorizer.transform(x_train['teacher_prefix'].values)
9 test_teacher_one_hot = vectorizer.transform(x_test['teacher_prefix'].values)
10 cv_teacher_one_hot = vectorizer.transform(x_cv['teacher_prefix'].values.astype('U'))
11
12 print("Shape of Train matrix after one hot encoding ",train_teacher_one_hot.shape)
13 print("Shape of Test matrix after one hot encoding ",test_teacher_one_hot.shape)
14 print("Shape of CV matrix after one hot encoding ",cv_teacher_one_hot.shape)

```

['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']  
 Shape of Train matrix after one hot encoding (49041, 5)  
 Shape of Test matrix after one hot encoding (36052, 5)  
 Shape of CV matrix after one hot encoding (24155, 5)

## 1.5.2 Vectorizing Text data

### 1.5.2.1 Bag of words

Bi-Grams of minimum frequency of words 10 (min\_df = 10) of 5000 features (max\_features = 5000)

In [47]:

```

1 # We are considering only the words which appeared in at least 10 documents
2 vectorizer = CountVectorizer(ngram_range = (2,2), max_features = 5000, min_df = 10)
3 vectorizer.fit(train_preprocessed_essays)
4
5 train_text_bow = vectorizer.transform(train_preprocessed_essays)
6 print("Shape of matrix after one hot encoding ",train_text_bow.shape)

```

Shape of matrix after one hot encoding (49041, 5000)

In [48]:

```

1 #Vectorizing Test Data
2 # you can vectorize the title also
3 # before you vectorize the title make sure you preprocess it
4 test_text_bow = vectorizer.transform(test_preprocessed_essays)
5 print("Shape of matrix after one hot encoding ",test_text_bow.shape)

```

Shape of matrix after one hot encoding (36052, 5000)

In [49]:

```

1 # Vectrozing CV Data
2 cv_text_bow = vectorizer.transform(cv_preprocessed_essays)
3 print("Shape of matrix after one hot encoding ",cv_text_bow.shape)

```

Shape of matrix after one hot encoding (24155, 5000)

## Project Title BOW

In [51]:

```

1 # We are considering only the words which appeared in at least 10 documents()
2 vectorizer = CountVectorizer(ngram_range = (2,2), max_features = 5000, min_d
3 vectorizer.fit(train_preprocessed_titles)
4
5 train_titles_bow = vectorizer.transform(train_preprocessed_titles)
6 print("Shape of matrix after one hot encoding ",train_titles_bow.shape)

```

Shape of matrix after one hot encoding (49041, 1670)

In [52]:

```

1 #Vectorizing Test Data
2 test_titles_bow = vectorizer.transform(test_preprocessed_titles)
3 print("Shape of matrix after one hot encoding ",test_titles_bow.shape)

```

Shape of matrix after one hot encoding (36052, 1670)

In [53]:

```

1 #Vectrizing CV Data
2 cv_titles_bow = vectorizer.transform(cv_preprocessed_titles)
3 print("Shape of matrix after one hot encoding ",cv_titles_bow.shape)

```

Shape of matrix after one hot encoding (24155, 1670)

### 1.5.2.2 TFIDF vectorizer

Here also we are considering the Bi-Grams of word frequency of 10 words(min\_df = 10) and Maximum features 5000

In [55]:

```

1 from sklearn.feature_extraction.text import TfidfVectorizer
2 vectorizer = TfidfVectorizer(min_df=10, max_features = 5000, ngram_range = (
3 vectorizer.fit(train_preprocessed_essays)
4
5 train_text_tfidf = vectorizer.transform(train_preprocessed_essays)
6 print("Shape of matrix after one hot encoding ",train_text_tfidf.shape)

```

Shape of matrix after one hot encoding (49041, 5000)

In [56]:

```

1 test_text_tfidf = vectorizer.transform(test_preprocessed_essays)
2 print("Shape of matrix after one hot encoding ",test_text_tfidf.shape)

```

Shape of matrix after one hot encoding (36052, 5000)

In [57]:

```

1 cv_text_tfidf = vectorizer.transform(cv_preprocessed_essays)
2 print("Shape of matrix after one hot encoding ",cv_text_tfidf.shape)

```

Shape of matrix after one hot encoding (24155, 5000)

### Project Titles

```
In [58]: 1 vectorizer = TfidfVectorizer(min_df=10, max_features = 5000, ngram_range = (2, 3))
          2 vectorizer.fit(train_preprocessed_titles)
          3
          4 train_title_tfidf = vectorizer.transform(train_preprocessed_titles)
          5 print("Shape of matrix after one hot encoding ",train_title_tfidf.shape)
```

Shape of matrix after one hot encoding (49041, 1670)

```
In [59]: 1 test_title_tfidf = vectorizer.transform(test_preprocessed_titles)
          2 print("Shape of matrix after one hot encoding ",test_title_tfidf.shape)
```

Shape of matrix after one hot encoding (36052, 1670)

```
In [60]: 1 cv_title_tfidf = vectorizer.transform(cv_preprocessed_titles)
          2 print("Shape of matrix after one hot encoding ",cv_title_tfidf.shape)
```

Shape of matrix after one hot encoding (24155, 1670)

### 1.5.2.3 Using Pretrained Models: Avg W2V

In [61]:

```
'1
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    4 print ("Loading Glove Model")
    5 f = open(gloveFile, 'r', encoding="utf8")
    6 model = {}
    7 for line in tqdm(f):
        8     splitLine = line.split()
        9     word = splitLine[0]
    10     embedding = np.array([float(val) for val in splitLine[1:]])
    11     model[word] = embedding
    12 print ("Done.",len(model)," words loaded!")
    13 return model
model = loadGloveModel('glove.42B.300d.txt')
15
#16=====
017: Output:
18
19 Loading Glove Model
20 1917495it [06:32, 4879.69it/s]
21 Done. 1917495 words loaded!
22
#23=====
24
25 words = []
26 for i in preproc_texts:
27     words.extend(i.split(' '))
28
29 for i in preproc_titles:
30     words.extend(i.split(' '))
31 print("all the words in the coupus", len(words))
32 words = set(words)
33 print("the unique words in the coupus", len(words))
34
35 inter_words = set(model.keys()).intersection(words)
36 print("The number of words that are present in both glove vectors and our coupus"
37       , len(inter_words), "(" ,np.round(len(inter_words)/len(words)*100,3), "%")
38
39 words_courpus = {}
40 words_glove = set(model.keys())
41 for i in words:
42     if i in words_glove:
43         words_courpus[i] = model[i]
44 print("word 2 vec length", len(words_courpus))
45
46
#47: Saving variables into pickle files python: http://www.jessicayung.com/how-to
48
49 import pickle
50 with open('glove_vectors', 'wb') as f:
51     pickle.dump(words_courpus, f)
52
53
'54
```

Out[61]: '\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084

```

039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGloveModel(glo
veFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,'r', enco
ding="utf8")\n    model = {} \n    for line in tqdm(f):\n        splitLine = lin
e.split()\n        word = splitLine[0]\n        embedding = np.array([float(va
l) for val in splitLine[1:]])\n        model[word] = embedding\n        print ("Don
e.",len(model)," words loaded!")\n    return model\nmodel = loadGloveModel('\\gl
ove.42B.300d.txt')\n\n# =====\nOutput:\n    \nLoading G
love Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495 words loaded!\n\n# =
===== \nwords = []\nfor i in preproc_texts:\n    word
s.extend(i.split(' '))\nfor i in preproc_titles:\n    words.extend(i.spli
t(' '))\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(mode
l.keys()).intersection(words)\nprint("The number of words that are present in b
oth glove vectors and our coupus", len(inter_words),",np.round(len(inte
r_words)/len(words)*100,3),%)")\n\nwords_courpus = {}\nwords_glove = set(mode
l.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# stronging va
riables 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 op
en('\\glove_vectors\\', 'wb') as f:\n    pickle.dump(words_courpus, f)\n\n"

```

In [71]:

```

1 # stronging variables into pickle files python: http://www.jessicayung.com/h
2 # make sure you have the glove_vectors file
3 with open('glove_vectors', 'rb') as f:
4     model = pickle.load(f)
5     glove_words = set(model.keys())

```

In [72]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 train_avg_w2v_vectors = [] # the avg-w2v for each sentence/review is stored
4 for sentence in tqdm(train_preprocessed_essays): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     cnt_words = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if word in glove_words:
9             vector += model[word]
10            cnt_words += 1
11    if cnt_words != 0:
12        vector /= cnt_words
13    train_avg_w2v_vectors.append(vector)
14
15 print(len(train_avg_w2v_vectors))
16 print(len(train_avg_w2v_vectors[0]))

```

100% |  
| 49041/49041 [00:21<00:00, 2250.66it/s]

49041

300

In [73]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 test_avg_w2v_vectors = [] # the avg-w2v for each sentence/review is stored
4 for sentence in tqdm(test_preprocessed_essays): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     cnt_words = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if word in glove_words:
9             vector += model[word]
10            cnt_words += 1
11    if cnt_words != 0:
12        vector /= cnt_words
13    test_avg_w2v_vectors.append(vector)
14
15 print(len(test_avg_w2v_vectors))
16 print(len(test_avg_w2v_vectors[0]))

```

100% |  
| 36052/36052 [00:16<00:00, 2137.89it/s]

36052  
300

In [74]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 cv_avg_w2v_vectors = [] # the avg-w2v for each sentence/review is stored in
4 for sentence in tqdm(cv_preprocessed_essays): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     cnt_words = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if word in glove_words:
9             vector += model[word]
10            cnt_words += 1
11    if cnt_words != 0:
12        vector /= cnt_words
13    cv_avg_w2v_vectors.append(vector)
14
15 print(len(cv_avg_w2v_vectors))
16 print(len(cv_avg_w2v_vectors[0]))

```

100% |  
| 24155/24155 [00:11<00:00, 2148.21it/s]

24155  
300

## AVG\_W2V Project\_Titles

In [75]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 train_title_avg_w2v_vectors = [] # the avg-w2v for each sentence/review is
4 for sentence in tqdm(train_preprocessed_titles): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     cnt_words = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if word in glove_words:
9             vector += model[word]
10            cnt_words += 1
11    if cnt_words != 0:
12        vector /= cnt_words
13    train_title_avg_w2v_vectors.append(vector)
14
15 print(len(train_title_avg_w2v_vectors))
16 print(len(train_title_avg_w2v_vectors[0]))

```

100% |  
49041/49041 [00:01<00:00, 41565.96it/s]

49041  
300

In [76]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 test_title_avg_w2v_vectors = [] # the avg-w2v for each sentence/review is s
4 for sentence in tqdm(test_preprocessed_titles): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     cnt_words = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if word in glove_words:
9             vector += model[word]
10            cnt_words += 1
11    if cnt_words != 0:
12        vector /= cnt_words
13    test_title_avg_w2v_vectors.append(vector)
14
15 print(len(test_title_avg_w2v_vectors))
16 print(len(test_title_avg_w2v_vectors[0]))

```

100% |  
36052/36052 [00:00<00:00, 39931.04it/s]

36052  
300

In [77]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 cv_title_avg_w2v_vectors = [] # the avg-w2v for each sentence/review is stored here
4 for sentence in tqdm(cv_preprocessed_titles): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     cnt_words = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if word in glove_words:
9             vector += model[word]
10            cnt_words += 1
11    if cnt_words != 0:
12        vector /= cnt_words
13    cv_title_avg_w2v_vectors.append(vector)
14
15 print(len(cv_title_avg_w2v_vectors))
16 print(len(cv_title_avg_w2v_vectors[0]))

```

100% |  
24155/24155 [00:00<00:00, 40063.13it/s]

24155  
300

### 1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [78]:

```

1 # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
2 tfidf_model = TfidfVectorizer()
3 tfidf_model.fit(train_preprocessed_essays)
4 # we are converting a dictionary with word as a key, and the idf as a value
5 dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
6 tfidf_words = set(tfidf_model.get_feature_names())

```

In [79]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 train_essay_tfidf_w2v_vectors = [] # the avg-w2v for each sentence/review is
4 for sentence in tqdm(train_preprocessed_essays): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     tf_idf_weight = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if (word in glove_words) and (word in tfidf_words):
9             vec = model[word] # getting the vector for each word
10            # here we are multiplying idf value(dictionary[word]) and the tf
11            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
12            vector += (vec * tf_idf) # calculating tfidf weighted w2v
13            tf_idf_weight += tf_idf
14        if tf_idf_weight != 0:
15            vector /= tf_idf_weight
16        train_essay_tfidf_w2v_vectors.append(vector)
17
18 print(len(train_essay_tfidf_w2v_vectors))
19 print(len(train_essay_tfidf_w2v_vectors[0]))

```

100% | 49041/49041 [02:35<00:00, 315.17it/s]

49041

300

In [80]:

```

1 # Similarly you can vectorize for title also
2 # average Word2Vec
3 # compute average word2vec for each review.
4 test_essay_tfidf_w2v_vectors = [] # the avg-w2v for each sentence/review is
5 for sentence in tqdm(test_preprocessed_essays): # for each review/sentence
6     vector = np.zeros(300) # as word vectors are of zero length
7     tf_idf_weight = 0; # num of words with a valid vector in the sentence/review
8     for word in sentence.split(): # for each word in a review/sentence
9         if (word in glove_words) and (word in tfidf_words):
10            vec = model[word] # getting the vector for each word
11            # here we are multiplying idf value(dictionary[word]) and the tf
12            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
13            vector += (vec * tf_idf) # calculating tfidf weighted w2v
14            tf_idf_weight += tf_idf
15        if tf_idf_weight != 0:
16            vector /= tf_idf_weight
17        test_essay_tfidf_w2v_vectors.append(vector)
18
19 print(len(test_essay_tfidf_w2v_vectors))
20 print(len(test_essay_tfidf_w2v_vectors[0]))

```

100% | 36052/36052 [01:52<00:00, 319.78it/s]

36052

300

In [81]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 cv_essay_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
4 for sentence in tqdm(cv_preprocessed_essays): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     tf_idf_weight = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if (word in glove_words) and (word in tfidf_words):
9             vec = model[word] # getting the vector for each word
10            # here we are multiplying idf value(dictionary[word]) and the tf
11            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
12            vector += (vec * tf_idf) # calculating tfidf weighted w2v
13            tf_idf_weight += tf_idf
14    if tf_idf_weight != 0:
15        vector /= tf_idf_weight
16    cv_essay_tfidf_w2v_vectors.append(vector)
17
18 print(len(cv_essay_tfidf_w2v_vectors))
19 print(cv_essay_tfidf_w2v_vectors[0]))

```

100% |  
| 24155/24155 [01:15<00:00, 320.56it/s]

24155

300

## Project\_titles

In [82]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 train_title_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
4 for sentence in tqdm(train_preprocessed_titles): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     tf_idf_weight = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if (word in glove_words) and (word in tfidf_words):
9             vec = model[word] # getting the vector for each word
10            # here we are multiplying idf value(dictionary[word]) and the tf
11            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
12            vector += (vec * tf_idf) # calculating tfidf weighted w2v
13            tf_idf_weight += tf_idf
14    if tf_idf_weight != 0:
15        vector /= tf_idf_weight
16    train_title_tfidf_w2v_vectors.append(vector)
17
18 print(len(train_title_tfidf_w2v_vectors))
19 print(len(train_title_tfidf_w2v_vectors[0]))

```

100% |  
| 49041/49041 [00:02<00:00, 18621.25it/s]

49041

300

In [83]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 test_title_tfidf_w2v_vectors = [] # the avg-w2v for each sentence/review is
4 for sentence in tqdm(test_preprocessed_titles): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     tf_idf_weight = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if (word in glove_words) and (word in tfidf_words):
9             vec = model[word] # getting the vector for each word
10            # here we are multiplying idf value(dictionary[word]) and the tf
11            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
12            vector += (vec * tf_idf) # calculating tfidf weighted w2v
13            tf_idf_weight += tf_idf
14        if tf_idf_weight != 0:
15            vector /= tf_idf_weight
16        test_title_tfidf_w2v_vectors.append(vector)
17
18 print(len(test_title_tfidf_w2v_vectors))
19 print(test_title_tfidf_w2v_vectors[0])

```

100%

36052/36052 [00:02&lt;00:00, 17461.34it/s]

36052

300

In [84]:

```

1 # average Word2Vec
2 # compute average word2vec for each review.
3 cv_title_tfidf_w2v_vectors = [] # the avg-w2v for each sentence/review is
4 for sentence in tqdm(cv_preprocessed_titles): # for each review/sentence
5     vector = np.zeros(300) # as word vectors are of zero length
6     tf_idf_weight = 0; # num of words with a valid vector in the sentence/review
7     for word in sentence.split(): # for each word in a review/sentence
8         if (word in glove_words) and (word in tfidf_words):
9             vec = model[word] # getting the vector for each word
10            # here we are multiplying idf value(dictionary[word]) and the tf
11            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
12            vector += (vec * tf_idf) # calculating tfidf weighted w2v
13            tf_idf_weight += tf_idf
14        if tf_idf_weight != 0:
15            vector /= tf_idf_weight
16        cv_title_tfidf_w2v_vectors.append(vector)
17
18 print(len(cv_title_tfidf_w2v_vectors))
19 print(cv_title_tfidf_w2v_vectors[0])

```

100%

24155/24155 [00:01&lt;00:00, 18427.66it/s]

24155

300

### 1.5.3 Vectorizing Numerical features

In [85]:

```

1 price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'})
2 project_data = pd.merge(project_data, price_data, on='id', how='left')
3 print(price_data.head())
4 #print(project_data.columns)
5 print(x_train.columns)

```

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21
2	p000003	298.97	4
3	p000004	1113.69	98
4	p000005	485.99	8

Index(['Unnamed: 0', 'id', 'teacher\_id', 'teacher\_prefix', 'school\_state', 'project\_submitted\_datetime', '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'],  
 dtype='object')

In [86]:

```

1 # - quantity : numerical (optional)
2 # - teacher_number_of_previously_posted_projects : numerical
3 # - price : numerical
4 x_train = pd.merge(x_train, price_data, on = "id", how = "left")
5 #print(x_train.columns)
6 x_test = pd.merge(x_test, price_data, on = "id", how = "left")
7 x_cv = pd.merge(x_cv, price_data, on = "id", how = "left")

```

## Standardize Price

In [87]:

```

1 # check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
2 # standardization sklearn: https://scikit-learn.org/stable/modules/generated
3 from sklearn.preprocessing import StandardScaler
4
5 # price_standardized = standardScalar.fit(project_data['price'].values)
6 # this will rise the error
7 # ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03
8 # Reshape your data either using array.reshape(-1, 1)
9
10 price_scalar = StandardScaler()
11
12 price_scalar.fit(x_train['price'].values.reshape(-1,1)) # finding the mean a
13 print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_
14
15 # Now standardize the data with above maen and variance.
16 train_price_standar = price_scalar.transform(x_train['price'].values.reshape(
17 train_price_standar

```

Mean : 297.1335653840664, Standard deviation : 363.28872800136634

Out[87]:

```
array([[[-0.40503201],
       [ 0.38780844],
       [-0.66273338],
       ...,
       [-0.64365764],
       [-0.19101491],
       [ 0.99591979]])
```

In [88]:

```

1 price_scalar.fit(x_test['price'].values.reshape(-1,1)) # finding the mean an
2 print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_
3
4 # Now standardize the data with above maen and variance.
5 test_price_standar = price_scalar.transform(x_test['price'].values.reshape(
6 test_price_standar

```

Mean : 300.7768501054033, Standard deviation : 382.6699753150988

Out[88]:

```
array([[ 0.05485967],
       [-0.77266801],
       [ 0.62508993],
       ...,
       [-0.77042065],
       [ 0.15630479],
       [-0.17139273]])
```

```
In [89]: 1 price_scalar.fit(x_cv['price'].values.reshape(-1,1)) # finding the mean and
2 print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_
3
4 # Now standardize the data with above maen and variance.
5 cv_price_standar = price_scalar.transform(x_cv['price'].values.reshape(-1, 1
6 test_price_standar
```

Mean : 296.1543266404471, Standard deviation : 352.5491742990371

```
Out[89]: array([[ 0.05485967],
 [-0.77266801],
 [ 0.62508993],
 ...,
 [-0.77042065],
 [ 0.15630479],
 [-0.17139273]])
```

```
In [90]: 1 print(train_price_standar.shape, y_train.shape)
2 print(test_price_standar.shape, y_test.shape)
3 print(cv_price_standar.shape, y_cv.shape)
```

(49041, 1) (49041,)  
(36052, 1) (36052,)  
(24155, 1) (24155,)

## Standardize Teacher previously posted Projects

```
In [91]: 1 warnings.filterwarnings("ignore")
2 price_scalar.fit(x_train['teacher_number_of_previously_posted_projects'].val
3 print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_
4
5 # Now standardize the data with above maen and variance.
6 train_prev_proj_standar = price_scalar.transform(x_train['teacher_number_of_7
train_prev_proj_standar
```

Mean : 11.093656328378296, Standard deviation : 27.627970646902444

```
Out[91]: array([[ 0.03280529],
 [-0.40153714],
 [-0.29295153],
 ...,
 [-0.32914673],
 [-0.29295153],
 [ 1.29963739]])
```

```
In [92]: 1 price_scalar.fit(x_test['teacher_number_of_previously_posted_projects'].values)
2 print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_
3
4 # Now standardize the data with above mean and variance.
5 test_prev_proj_standar = price_scalar.transform(x_test['teacher_number_of_pr
6 test_prev_proj_standar
```

Mean : 11.299123488294686, Standard deviation : 28.185609983236137

```
Out[92]: array([[-0.22348722],
 [-0.3654036 ],
 [-0.3654036 ],
 ...,
 [ 2.40196599],
 [-0.3654036 ],
 [-0.4008827 ]])
```

```
In [93]: 1 price_scalar.fit(x_cv['teacher_number_of_previously_posted_projects'].values
2 print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_
3
4 # Now standardize the data with above mean and variance.
5 cv_prev_proj_standar = price_scalar.transform(x_cv['teacher_number_of_previo
6 cv_prev_proj_standar
```

Mean : 11.056137445663424, Standard deviation : 27.46187050275317

```
Out[93]: array([[-0.40259958],
 [ 0.83548069],
 [ 0.21644056],
 ...,
 [-0.40259958],
 [ 0.14361231],
 [-0.25694307]])
```

```
In [94]: 1 print(train_prev_proj_standar.shape, y_train.shape)
2 print(test_prev_proj_standar.shape, y_test.shape)
3 print(cv_prev_proj_standar.shape, y_cv.shape)
```

(49041, 1) (49041,)  
(36052, 1) (36052,)  
(24155, 1) (24155,)

## Standardize Quantity

```
In [95]: 1 price_scalar.fit(x_train['quantity'].values.reshape(-1,1)) # finding the mean
2 print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_
3
4 # Now standardize the data with above maen and variance.
5 train_quantity_standar = price_scalar.transform(x_train['quantity'].values.r
6 train_quantity_standar
7
```

Mean : 17.10701249974511, Standard deviation : 27.48278828673773

```
Out[95]: array([[-0.25859867],
 [-0.54968995],
 [-0.18582585],
 ...,
 [-0.22221226],
 [-0.29498508],
 [-0.47691713]])
```

```
In [96]: 1 price_scalar.fit(x_test['quantity'].values.reshape(-1,1)) # finding the mean
2 print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_
3
4 # Now standardize the data with above maen and variance.
5 test_quantity_standar = price_scalar.transform(x_test['quantity'].values.res
6 test_quantity_standar
```

Mean : 16.771219349828026, Standard deviation : 24.416340622026798

```
Out[96]: array([[-0.11349855],
 [ 0.13223852],
 [-0.60497269],
 ...,
 [ 2.01622272],
 [-0.44114798],
 [-0.27732327]])
```

```
In [97]: 1 price_scalar.fit(x_cv['quantity'].values.reshape(-1,1)) # finding the mean a
2 print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_
3
4 # Now standardize the data with above maen and variance.
5 cv_quantity_standar = price_scalar.transform(x_cv['quantity'].values.reshape
6 cv_quantity_standar
```

Mean : 16.968660732767542, Standard deviation : 26.023272927346344

```
Out[97]: array([[-0.26778571],
 [-0.61362999],
 [ 0.07805856],
 ...,
 [-0.45992142],
 [ 1.3077271 ],
 [ 0.03963142]])
```

In [98]:

```

1 print(train_quantity_standar.shape, y_train.shape)
2 print(test_quantity_standar.shape, y_test.shape)
3 print(cv_quantity_standar.shape, y_cv.shape)

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

Computing Sentiment Scores

In [99]:

```

1 import nltk
2 from nltk.sentiment.vader import SentimentIntensityAnalyzer
3
4 # import nltk
5 #nltk.download('vader_lexicon')
6
7 sid = SentimentIntensityAnalyzer()
8
9 for_sentiment = 'a person is a person no matter how small dr seuss i teach t
10 for learning my students learn in many different ways using all of our sense
11 of techniques to help all my students succeed students in my class come from
12 for wonderful sharing of experiences and cultures including native americans
13 learners which can be seen through collaborative student project based learn
14 in my class love to work with hands on materials and have many different opp
15 mastered having the social skills to work cooperatively with friends is a cr
16 montana is the perfect place to learn about agriculture and nutrition my stu
17 in the early childhood classroom i have had several kids ask me can we try c
18 and create common core cooking lessons where we learn important math and wri
19 food for snack time my students will have a grounded appreciation for the wo
20 of where the ingredients came from as well as how it is healthy for their bo
21 nutrition and agricultural cooking recipes by having us peel our own apples
22 and mix up healthy plants from our classroom garden in the spring we will al
23 shared with families students will gain math and literature skills as well a
24 nannan'
25 ss = sid.polarity_scores(for_sentiment)
26
27 for k in ss:
28     print('{0}: {1}, '.format(k, ss[k]), end=' ')
29
30 # we can use these 4 things as features/attributes (neg, neu, pos, compound)
31 # neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,

## Assignment 5: Logistic Regression

1. [Task-1] Logistic Regression(either SGDClassifier with log loss, or LogisticRegression) on these feature sets
  - Set 1: categorical, numerical features + project\_title(BOW) + preprocessed\_eassay ('BOW with bi-grams` with `min\_df=10` and `max\_features=5000`)

- **Set 2:** categorical, numerical features + project\_title(TFIDF)+ preprocessed\_eassay ('TFIDF with bi-grams` with `min\_df=10` and `max\_features=5000`)
- **Set 3:** categorical, numerical features + project\_title(AVG W2V)+ preprocessed\_eassay (AVG W2V)
- **Set 4:** categorical, numerical features + project\_title(TFIDF W2V)+ preprocessed\_essay (TFIDF W2V)

## 2. Hyper parameter tuning (find best hyper parameters corresponding the algorithm that you choose)

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

## 3. Representation of results

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

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

 Along with plotting ROC curve, you need to print the confusion matrix (<https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/>) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

 (<https://seaborn.pydata.org/generated/seaborn.heatmap.html>)

## 4. [Task-2] Apply Logistic Regression on the below feature set **Set 5** by finding the best hyper parameter as suggested in step 2 and step 3.

### 5. Consider these set of features **Set 5** :

- **school\_state** : categorical data
- **clean\_categories** : categorical data
- **clean\_subcategories** : categorical data
- **project\_grade\_category** : categorical data
- **teacher\_prefix** : categorical data
- **quantity** : numerical data
- **teacher\_number\_of\_previously\_posted\_projects** : numerical data
- **price** : numerical data
- **sentiment score's of each of the essay** : numerical data
- **number of words in the title** : numerical data
- **number of words in the combine essays** : numerical data

And apply the Logistic regression on these features by finding the best hyper parameter as suggested in step 2 and step 3

## 6. Conclusion

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



### Note: Data Leakage

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

## LOGISTIC REGRESSION

### 2.1 Logistic Regression On Set-1

#### Merging all Categorical and Numerical \_ SET-1 BOW Encoding

In [135]:

```

1 from scipy.sparse import hstack
2 # with the same hstack function we are concatenating a sparse matrix and a d
3 X_train1 = hstack((train_categories_one_hot,train_sub_categories_one_hot,tra
4                   train_teacher_one_hot,train_text_bow, train_titles_bow, tr
5                   train_prev_proj_standar, train_price_standar)).tocsr()
6 print(X_train1.shape, y_train.shape)
7 print(type(X_train1))
8

```

(49041, 6773) (49041,)  
<class 'scipy.sparse.csr.csr\_matrix'>

In [114]:

```

1 X_test1 = hstack((test_categories_one_hot,test_sub_categories_one_hot,test_s
2                   test_teacher_one_hot,test_text_bow, test_titles_bow, test_
3                   test_prev_proj_standar, test_price_standar)).tocsr()
4 print(X_test1.shape, y_test.shape)
5 print(type(X_test1))

```

(36052, 6773) (36052,)  
<class 'scipy.sparse.csr.csr\_matrix'>

```
In [115]: 1 X_cv1 = hstack((cv_categories_one_hot, cv_sub_categories_one_hot, cv_state_o  
2           cv_teacher_one_hot, cv_text_bow, cv_titles_bow, cv_quantit  
3           cv_prev_proj_standar, cv_price_standar)).tocsr()  
4 print(X_cv1.shape, y_cv.shape)  
5 print(type(X_cv1))
```

```
(24155, 6773) (24155,  
<class 'scipy.sparse.csr.csr_matrix'>
```

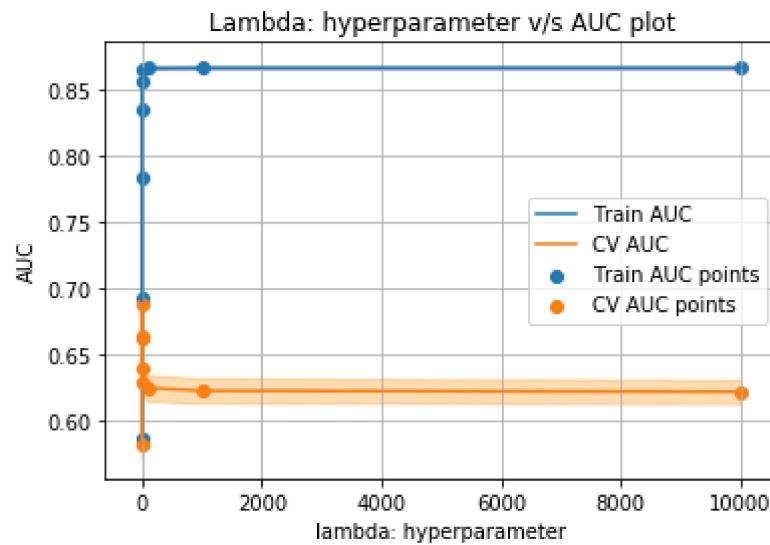
```
In [116]: 1 print(X_train1.shape, y_train.shape)
```

```
(49041, 6773) (49041,)
```

## Hyperparameter Tuning

```
In [124]: 1 from sklearn.model_selection import train_test_split  
2 from sklearn.grid_search import GridSearchCV  
3 from sklearn.model_selection import GridSearchCV  
4 #from sklearn.datasets import *  
5 from sklearn.linear_model import LogisticRegression
```

```
In [126]: 1 parameters = {'C':[10**4, 10**3, 10**2, 10**1, 10**0, 10**-1, 10**-2, 10**-3]
2
3 classifier = GridSearchCV(LogisticRegression(), parameters, cv= 10, scoring=
4
5 classifier.fit(X_train1, y_train)
6
7 train_auc= classifier.cv_results_['mean_train_score']
8 train_auc_std= classifier.cv_results_['std_train_score']
9 cv_auc = classifier.cv_results_['mean_test_score']
10 cv_auc_std= classifier.cv_results_['std_test_score']
11
12 plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/408403
13 plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
14
15
16 plt.plot(parameters['C'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/408403
17 plt.gca().fill_between(parameters['C'],cv_auc - cv_auc_std, cv_auc + cv_auc_s
18
19
20 plt.scatter(parameters['C'], train_auc, label='Train AUC points')
21 plt.scatter(parameters['C'], cv_auc, label='CV AUC points')
22
23
24 plt.legend()
25 plt.xlabel("hyperparameter")
26 plt.ylabel("Accuracy")
27 plt.title("Accuracy vs hyperparameter plot")
28 plt.grid()
29 plt.show()
```



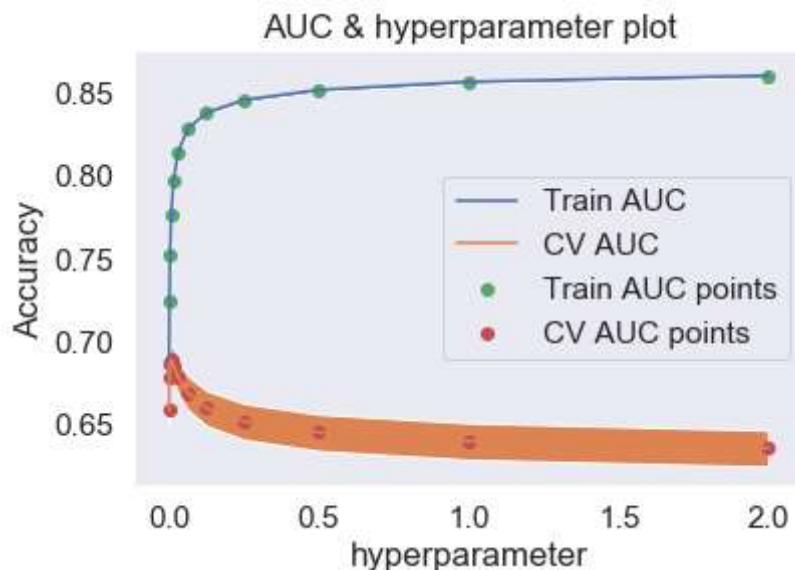
**As we can see Train and CV curve is almost linear from 500 to 10000. So we are using less C values, that might give best hyperparameter for our classifier**

In [160]:

```

1 parameters = {'C':[2, 1, 0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078
2
3 classifier = GridSearchCV(LogisticRegression(), parameters, cv= 10, scoring=
4
5 classifier.fit(X_train1, y_train)
6
7 train_auc= classifier.cv_results_['mean_train_score']
8 train_auc_std= classifier.cv_results_['std_train_score']
9 cv_auc = classifier.cv_results_['mean_test_score']
10 cv_auc_std= classifier.cv_results_['std_test_score']
11 #how to draw roc curve for logistic regression gridsearchcv
12 #https://stackoverflow.com/questions/48796282/how-to-visualize-dependence-of
13 #https://stackoverflow.com/questions/42894871/how-to-plot-multiple-roc-curve
14 #
15 plt.plot(parameters['C'], train_auc, label='Train AUC')
16 plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
17
18 plt.plot(parameters['C'], cv_auc, label='CV AUC')
19 plt.gca().fill_between(parameters['C'],cv_auc - cv_auc_std, cv_auc + cv_auc_s
20
21 plt.scatter(parameters['C'], train_auc, label='Train AUC points')
22 plt.scatter(parameters['C'], cv_auc, label='CV AUC points')
23
24
25 plt.legend()
26 plt.xlabel("hyperparameter")
27 plt.ylabel("Accuracy")
28 plt.title("AUC hyperparameter plot")
29 plt.grid()
30 plt.show()

```

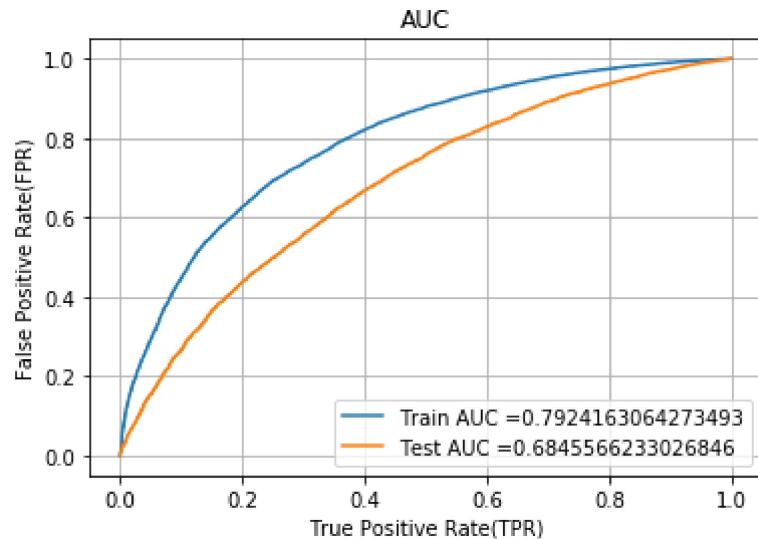


1

```
In [130]: 1 def batch_predict(clf, data):
2     # roc_auc_score(y_true, y_score) the 2nd parameter should be probability
3     # not the predicted outputs
4
5     y_data_pred = []
6     tr_loop = data.shape[0] - data.shape[0]//1000
7     # consider you X_tr shape is 49041, then your cr_Loop will be 49041 - 49
8     # in this for Loop we will iterate until the last 1000 multiplier
9     for i in range(0, tr_loop, 1000):
10         y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
11         # we will be predicting for the last data points
12     y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
13
14 return y_data_pred
```

```
In [145]: 1 best_c_1 = 0.015625
```

```
In [147]: 1 # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
2 from sklearn.metrics import roc_curve, auc
3
4 LGR_bow = LogisticRegression(C = best_c_1)
5
6 LGR_bow.fit(X_train1, y_train)
7 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
8
9
10 y_train_pred = batch_predict(LGR_bow, X_train1)
11 y_test_pred = batch_predict(LGR_bow, X_test1)
12
13 train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
14 test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
15
16 plt.plot(train_fpr, train_tpr, label="Train AUC =" + str(auc(train_fpr, train_tpr)))
17 plt.plot(test_fpr, test_tpr, label="Test AUC =" + str(auc(test_fpr, test_tpr)))
18 plt.legend()
19 plt.xlabel("True Positive Rate(TPR)")
20 plt.ylabel("False Positive Rate(FPR)")
21 plt.title("AUC")
22 plt.grid()
23 plt.show()
```



## Confusion Matrix

```
In [148]: 1 def predict(proba, threshold, fpr, tpr):
2
3     t = threshold[np.argmax(fpr*(1-tpr))]
4     print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold")
5     predictions = []
6     for i in proba:
7         if i>=t:
8             predictions.append(1)
9         else:
10            predictions.append(0)
11    return predictions
```

## Train Data

In [149]:

```
1 from sklearn.metrics import confusion_matrix
2 import seaborn as sea
3 print("Train confusion matrix")
4 print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_f
```

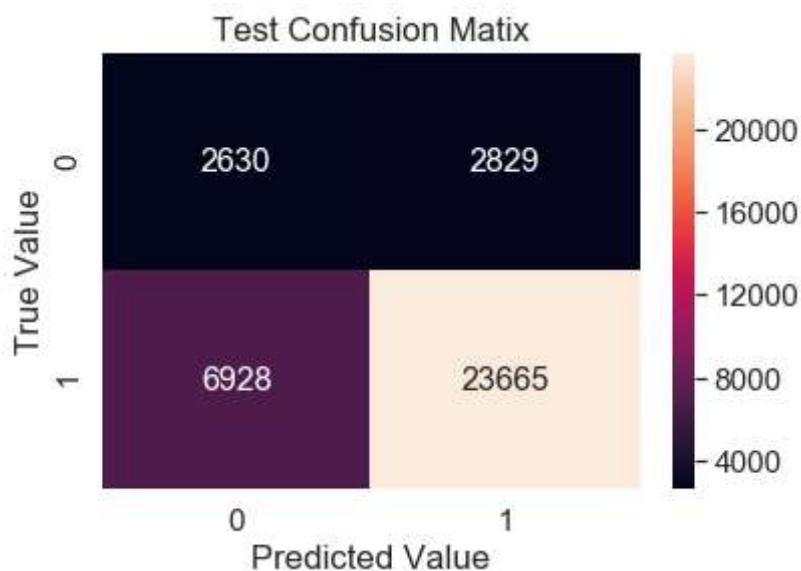
Train confusion matrix  
the maximum value of tpr\*(1-fpr) 0.25 for threshold 0.772  
[[ 3713 3713]  
[ 5064 36551]]

In [150]:

```
1 train_confusion_matrix = pd.DataFrame(confusion_matrix(y_test,predict(y_test,
2 test_f
3 sea.set(font_scale=1.4)
4 sea.heatmap(train_confusion_matrix, annot = True, annot_kws={"size":16}, fmt
5 plt.xlabel("Predicted Value")
6 plt.ylabel("True Value")
7 plt.title("Test Confusion Matix")
```

the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.808

Out[150]: Text(0.5,1,'Test Confusion Matix')



## Test Data

In [151]:

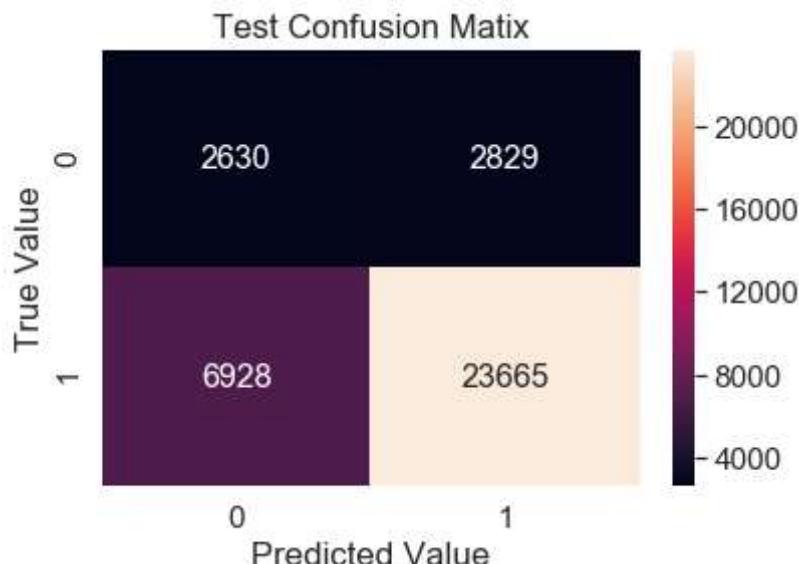
```
1 print("Test confusion matrix")
2 print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr,
```

Test confusion matrix  
the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.808  
[[ 2630 2829]  
[ 6928 23665]]

```
In [152]: 1 train_confusion_matrix = pd.DataFrame(confusion_matrix(y_test,predict(y_test))
2
3 sea.set(font_scale=1.4)
4 sea.heatmap(train_confusion_matrix, annot = True, annot_kws={"size":16}, fmt
5 plt.xlabel("Predicted Value")
6 plt.ylabel("True Value")
7 plt.title("Test Confusion Matix")
```

the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.808

Out[152]: Text(0.5,1,'Test Confusion Matix')



## 2.2 Merging all Categorical and Numerical \_ SET-2 TF-IDF Encoding

```
In [155]: 1 X_train2 = hstack((train_categories_one_hot,train_sub_categories_one_hot,tra
2
3
4 print(X_train2.shape, y_train.shape)
5 print(type(X_train2))
```

(49041, 6773) (49041,)  
<class 'scipy.sparse.csr.csr\_matrix'>

```
In [158]: 1 X_test2 = hstack((test_categories_one_hot,test_sub_categories_one_hot,test_s
2
3
4 print(X_test2.shape, y_test.shape)
5 print(type(X_test2))
```

(36052, 6773) (36052,)  
<class 'scipy.sparse.csr.csr\_matrix'>

```
In [159]: 1 X_cv2 = hstack((cv_categories_one_hot, cv_sub_categories_one_hot, cv_state_one
2           cv_teacher_one_hot, cv_text_tfidf, cv_title_tfidf, cv_quant
3           cv_prev_proj_standar, cv_price_standar)).tocsr()
4 print(X_cv2.shape, y_cv.shape)
5 print(type(X_cv2))
```

```
(24155, 6773) (24155,)
<class 'scipy.sparse.csr.csr_matrix'>
```

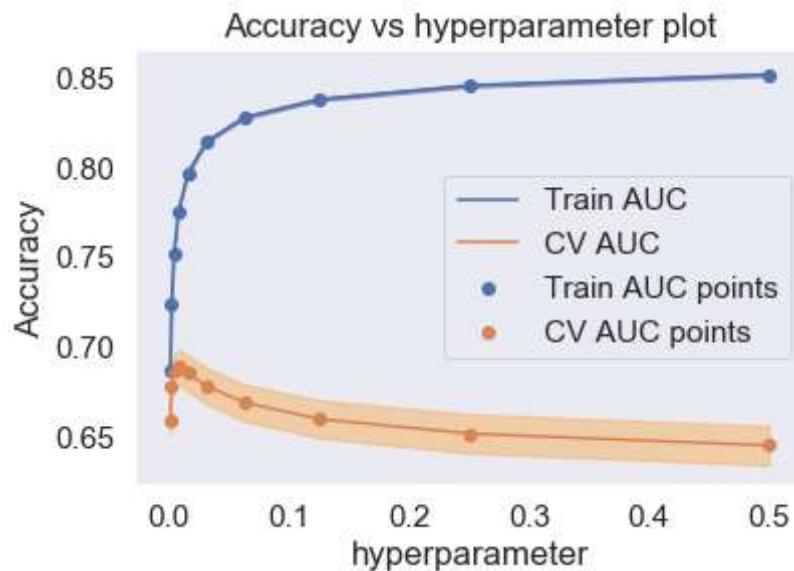
## Hyperparameter Tuning

In [161]:

```

1 parameters = {'C':[0.5, 0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078125, 0
2
3 classifier = GridSearchCV(LogisticRegression(), parameters, cv= 10, scoring=
4 classifier.fit(X_train1, y_train)
5
6 train_auc= classifier.cv_results_['mean_train_score']
7 train_auc_std= classifier.cv_results_['std_train_score']
8 cv_auc = classifier.cv_results_['mean_test_score']
9 cv_auc_std= classifier.cv_results_['std_test_score']
10
11 plt.plot(parameters['C'], train_auc, label='Train AUC')
12 plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
13
14 plt.plot(parameters['C'], cv_auc, label='CV AUC')
15 plt.gca().fill_between(parameters['C'],cv_auc - cv_auc_std, cv_auc + cv_auc_s
16
17 plt.scatter(parameters['C'], train_auc, label='Train AUC points')
18 plt.scatter(parameters['C'], cv_auc, label='CV AUC points')
19
20
21 plt.legend()
22 plt.xlabel("hyperparameter")
23 plt.ylabel("Accuracy")
24 plt.title("Accuracy vs hyperparameter plot")
25 plt.grid()
26 plt.show()

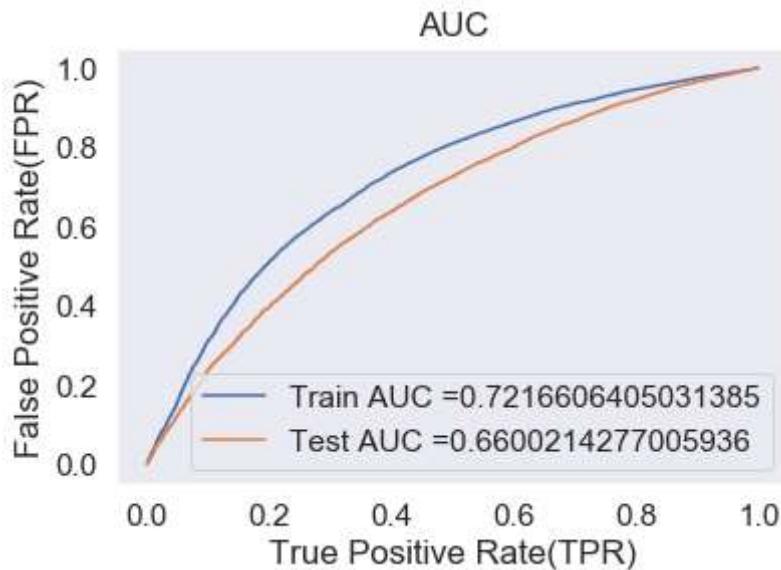
```



In [162]:

1 best\_c\_2 = 0.0039

```
In [164]: 1 # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html
2 from sklearn.metrics import roc_curve, auc
3
4 LGR_tfidf = LogisticRegression(C = best_c_2)
5
6 LGR_tfidf.fit(X_train2, y_train)
7 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates
# not the predicted outputs
8
9
10 y_train_pred = batch_predict(LGR_bow, X_train2)
11 y_test_pred = batch_predict(LGR_bow, X_test2)
12
13 train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
14 test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
15
16 plt.plot(train_fpr, train_tpr, label="Train AUC =" + str(auc(train_fpr, train_tpr)))
17 plt.plot(test_fpr, test_tpr, label="Test AUC =" + str(auc(test_fpr, test_tpr)))
18 plt.legend()
19 plt.xlabel("True Positive Rate(TPR)")
20 plt.ylabel("False Positive Rate(FPR)")
21 plt.title("AUC")
22 plt.grid()
23 plt.show()
```



## Train Confusing Matrix

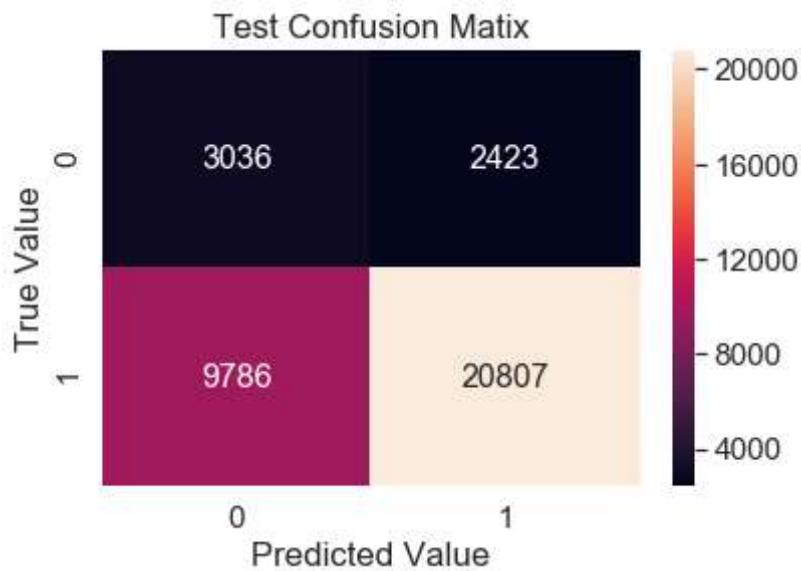
```
In [165]: 1 from sklearn.metrics import confusion_matrix
2 import seaborn as sea
3 print("Train confusion matrix")
4 print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_f
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.762
[[ 3713  3713]
 [ 7951 33664]]
```

```
In [166]: 1 train_confusion_matrix = pd.DataFrame(confusion_matrix(y_test,predict(y_test))
2
3 sea.set(font_scale=1.4)
4 sea.heatmap(train_confusion_matrix, annot = True, annot_kws={"size":16}, fmt
5 plt.xlabel("Predicted Value")
6 plt.ylabel("True Value")
7 plt.title("Test Confusion Matix")
```

the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.777

Out[166]: Text(0.5,1,'Test Confusion Matix')



## Test Confusion Matrix

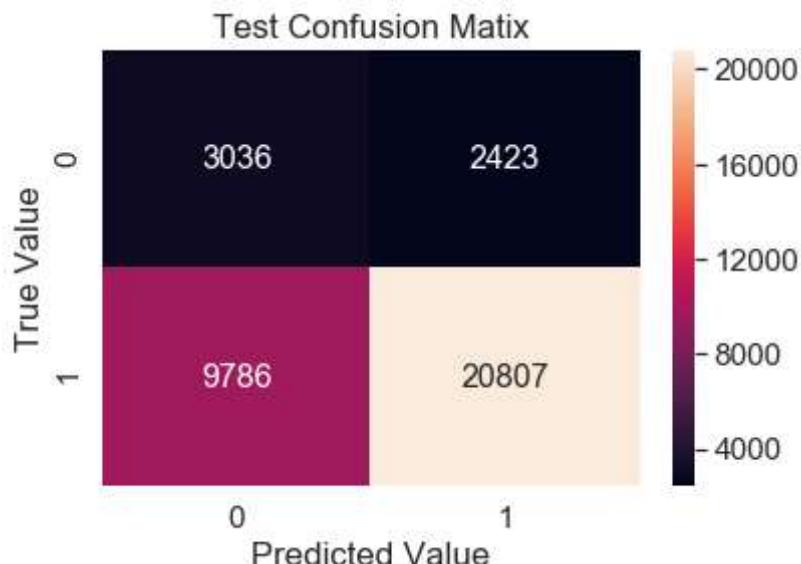
```
In [167]: 1 print("Test confusion matrix")
2 print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr,
```

Test confusion matrix  
the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.777  
[[ 3036 2423]  
 [ 9786 20807]]

```
In [168]: 1 train_confusion_matrix = pd.DataFrame(confusion_matrix(y_test,predict(y_test))
2
3 sea.set(font_scale=1.4)
4 sea.heatmap(train_confusion_matrix, annot = True, annot_kws={"size":16}, fmt
5 plt.xlabel("Predicted Value")
6 plt.ylabel("True Value")
7 plt.title("Test Confusion Matix")
```

the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.777

Out[168]: Text(0.5,1,'Test Confusion Matix')



In [ ]:

## 2.3 Merging all Categorical and Numerical \_ SET-3 AVG-W2V Encoding

```
In [169]: 1 from scipy.sparse import hstack
2 # with the same hstack function we are concatenating a sparse matrix and a d
3 X_train3 = hstack((train_categories_one_hot,train_sub_categories_one_hot,tra
4                   train_teacher_one_hot, train_title_avg_w2v_vectors, train_
5                   train_prev_proj_standar, train_price_standar)).tocsr()
6 print(X_train3.shape, y_train.shape)
7 print(type(X_train3))
```

(49041, 703) (49041,)  
<class 'scipy.sparse.csr.csr\_matrix'>

```
In [171]: 1 X_test3 = hstack((test_categories_one_hot,test_sub_categories_one_hot,test_s  
2           test_teacher_one_hot, test_title_avg_w2v_vectors, test_avg  
3           test_prev_proj_standar, test_price_standar)).tocsr()  
4 print(X_test3.shape, y_test.shape)  
5 print(type(X_test3))
```

```
(36052, 703) (36052,)  
<class 'scipy.sparse.csr.csr_matrix'>
```

```
In [172]: 1 X_cv3 = hstack((cv_categories_one_hot, cv_sub_categories_one_hot, cv_state_o  
2           cv_teacher_one_hot, cv_title_avg_w2v_vectors, cv_avg_w2v_v  
3           cv_prev_proj_standar, cv_price_standar)).tocsr()  
4 print(X_cv3.shape, y_cv.shape)  
5 print(type(X_cv3))
```

```
(24155, 703) (24155,)  
<class 'scipy.sparse.csr.csr_matrix'>
```

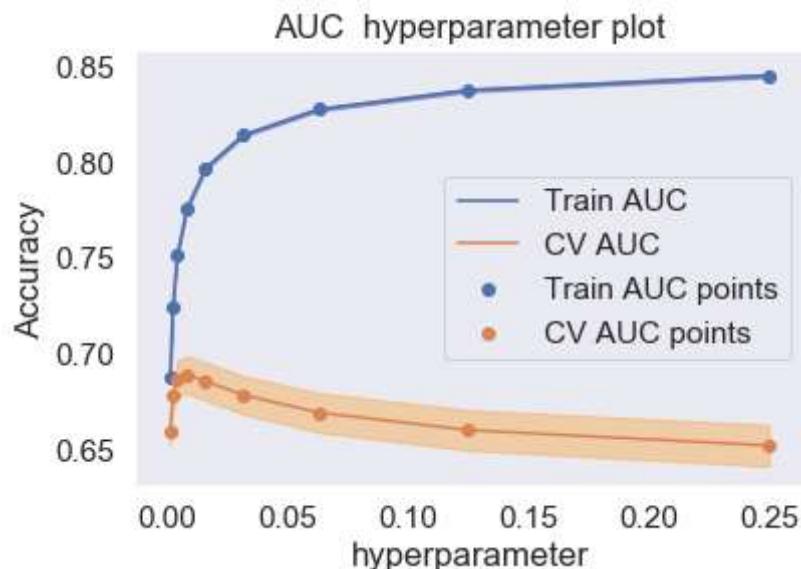
## Hyperparameter Tuning

In [174]:

```

1 parameters = {'C':[0.25, 0.125, 0.0625, 0.03125, 0.015625, 0.0078125, 0.0039
2
3 classifier = GridSearchCV(LogisticRegression(), parameters, cv= 10, scoring=
4 classifier.fit(X_train1, y_train)
5
6 train_auc= classifier.cv_results_['mean_train_score']
7 train_auc_std= classifier.cv_results_['std_train_score']
8 cv_auc = classifier.cv_results_['mean_test_score']
9 cv_auc_std= classifier.cv_results_['std_test_score']
10 #how to draw roc curve for Logistic regression gridsearchcv
11 #https://stackoverflow.com/questions/48796282/how-to-visualize-dependence-of
12 #https://stackoverflow.com/questions/42894871/how-to-plot-multiple-roc-curve
13 #
14 plt.plot(parameters['C'], train_auc, label='Train AUC')
15 plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
16
17 plt.plot(parameters['C'], cv_auc, label='CV AUC')
18 plt.gca().fill_between(parameters['C'],cv_auc - cv_auc_std, cv_auc + cv_auc_s
19
20 plt.scatter(parameters['C'], train_auc, label='Train AUC points')
21 plt.scatter(parameters['C'], cv_auc, label='CV AUC points')
22
23
24 plt.legend()
25 plt.xlabel("hyperparameter")
26 plt.ylabel("Accuracy")
27 plt.title("AUC hyperparameter plot")
28 plt.grid()
29 plt.show()

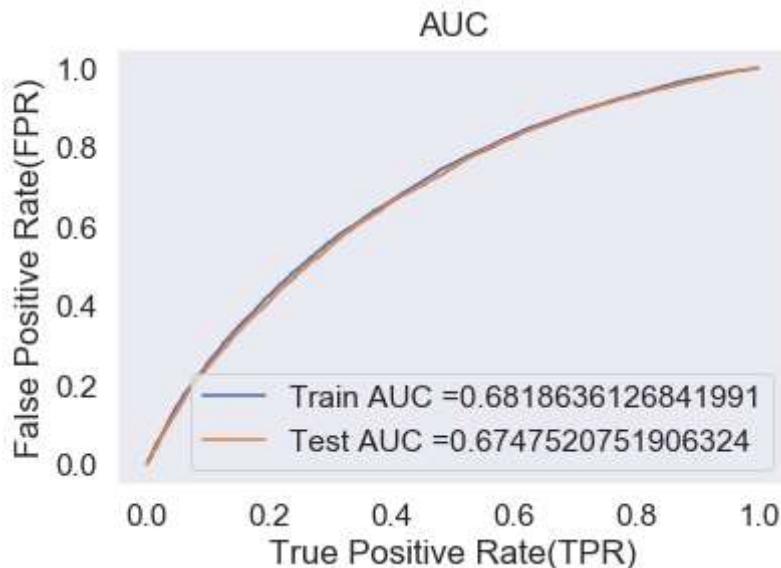
```



In [175]:

1 best\_c\_3 = 0.0039

```
In [176]: 1 # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve
2 from sklearn.metrics import roc_curve, auc
3
4 LGR_avgw2v = LogisticRegression(C = best_c_3)
5
6 LGR_avgw2v.fit(X_train3, y_train)
7 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
8 # not the predicted outputs
9
10 y_train_pred = batch_predict(LGR_avgw2v, X_train3)
11 y_test_pred = batch_predict(LGR_avgw2v, X_test3)
12
13 train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
14 test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
15
16 plt.plot(train_fpr, train_tpr, label="Train AUC =" + str(auc(train_fpr, train_tpr)))
17 plt.plot(test_fpr, test_tpr, label="Test AUC =" + str(auc(test_fpr, test_tpr)))
18 plt.legend()
19 plt.xlabel("True Positive Rate(TPR)")
20 plt.ylabel("False Positive Rate(FPR)")
21 plt.title("AUC")
22 plt.grid()
23 plt.show()
```



## Train Confusion Matrix

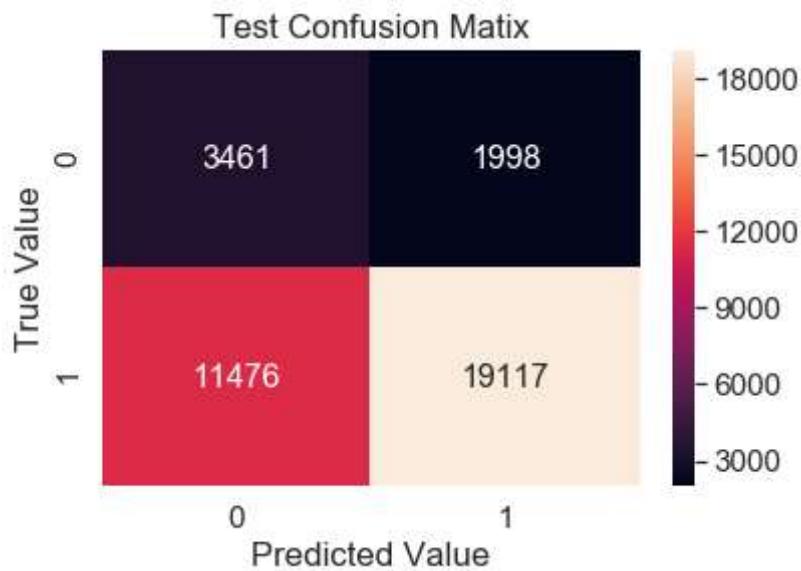
```
In [177]: 1 from sklearn.metrics import confusion_matrix
2 import seaborn as sea
3 print("Train confusion matrix")
4 print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_f
```

```
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.822
[[ 3713  3713]
 [10080 31535]]
```

```
In [178]: 1 train_confusion_matrix = pd.DataFrame(confusion_matrix(y_test,predict(y_test))
2
3 sea.set(font_scale=1.4)
4 sea.heatmap(train_confusion_matrix, annot = True, annot_kws={"size":16}, fmt
5 plt.xlabel("Predicted Value")
6 plt.ylabel("True Value")
7 plt.title("Test Confusion Matix")
```

the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.843

Out[178]: Text(0.5,1,'Test Confusion Matix')



## Test Confusion Matrix

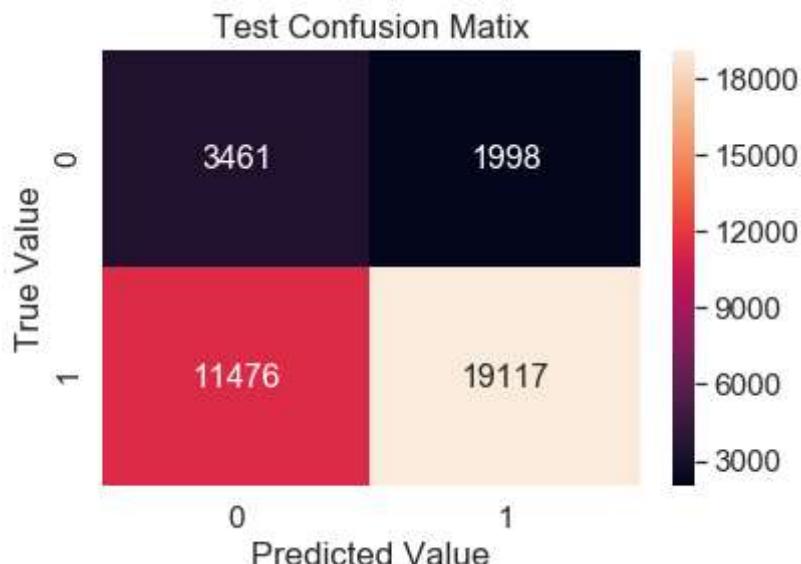
```
In [179]: 1 print("Test confusion matrix")
2 print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr,
```

Test confusion matrix  
the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.843  
[[ 3461 1998]  
 [11476 19117]]

```
In [180]: 1 train_confusion_matrix = pd.DataFrame(confusion_matrix(y_test,predict(y_test))
2
3 sea.set(font_scale=1.4)
4 sea.heatmap(train_confusion_matrix, annot = True, annot_kws={"size":16}, fmt
5 plt.xlabel("Predicted Value")
6 plt.ylabel("True Value")
7 plt.title("Test Confusion Matix")
```

the maximum value of tpr\*(1-fpr) 0.24999999161092998 for threshold 0.843

Out[180]: Text(0.5,1,'Test Confusion Matix')



## 2.4 Merging all Categorical and Numerical \_ SET-4 TFIDF-W2V Encoding

```
In [181]: 1 from scipy.sparse import hstack
2 # with the same hstack function we are concatenating a sparse matrix and a d
3 X_train4 = hstack((train_categories_one_hot,train_sub_categories_one_hot,tra
4                 train_teacher_one_hot, train_title_tfidf_w2v_vectors, tra
5                 train_quantity_standar, train_prev_proj_standar, train_pr
6 print(X_train4.shape, y_train.shape)
7 print(type(X_train4)) #train_title_tfidf_w2v_vectors train_essay_tfidf_w2v_v
```

(49041, 703) (49041,)  
<class 'scipy.sparse.csr.csr\_matrix'>

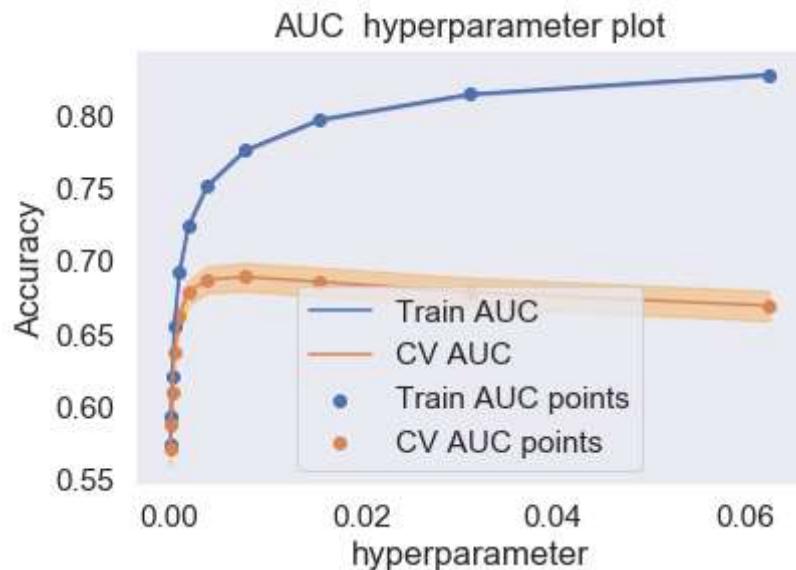
```
In [182]: 1 X_test4 = hstack((test_categories_one_hot,test_sub_categories_one_hot,test_s
2                         test_teacher_one_hot, test_title_tfidf_w2v_vectors, test_
3                         test_quantity_standar, test_prev_proj_standar, test_price
4 print(X_test4.shape, y_test.shape)
5 print(type(X_test4)) #train_title_tfidf_w2v_vectors train_essay_tfidf_w2v_ve
```

(36052, 703) (36052,)  
<class 'scipy.sparse.csr.csr\_matrix'>

```
In [183]: 1 X_cv4 = hstack((cv_categories_one_hot, cv_sub_categories_one_hot, cv_state_o  
2                      cv_teacher_one_hot, cv_title_tfidf_w2v_vectors, cv_essay_  
3                      cv_quantity_standar, cv_prev_proj_standar, cv_price_stand  
4 print(X_cv4.shape, y_cv.shape)  
5 print(type(X_cv4)) #train_title_tfidf_w2v_vectors train_essay_tfidf_w2v_vect  
  
(24155, 703) (24155,)  
<class 'scipy.sparse.csr.csr_matrix'>
```

## Hyperparameter Tuning

```
In [184]: 1 parameters = {'C':[0.0625, 0.03125, 0.015625, 0.0078125, 0.0039, 0.00195, 0.
2
3 classifier = GridSearchCV(LogisticRegression(), parameters, cv= 10, scoring=
4 classifier.fit(X_train1, y_train)
5
6 train_auc= classifier.cv_results_['mean_train_score']
7 train_auc_std= classifier.cv_results_['std_train_score']
8 cv_auc = classifier.cv_results_['mean_test_score']
9 cv_auc_std= classifier.cv_results_['std_test_score']
10
11 plt.plot(parameters['C'], train_auc, label='Train AUC')
12 plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc +
13
14 plt.plot(parameters['C'], cv_auc, label='CV AUC')
15 plt.gca().fill_between(parameters['C'],cv_auc - cv_auc_std, cv_auc + cv_auc_s
16
17 plt.scatter(parameters['C'], train_auc, label='Train AUC points')
18 plt.scatter(parameters['C'], cv_auc, label='CV AUC points')
19
20
21 plt.legend()
22 plt.xlabel("hyperparameter")
23 plt.ylabel("Accuracy")
24 plt.title("AUC hyperparameter plot")
25 plt.grid()
26 plt.show()
```



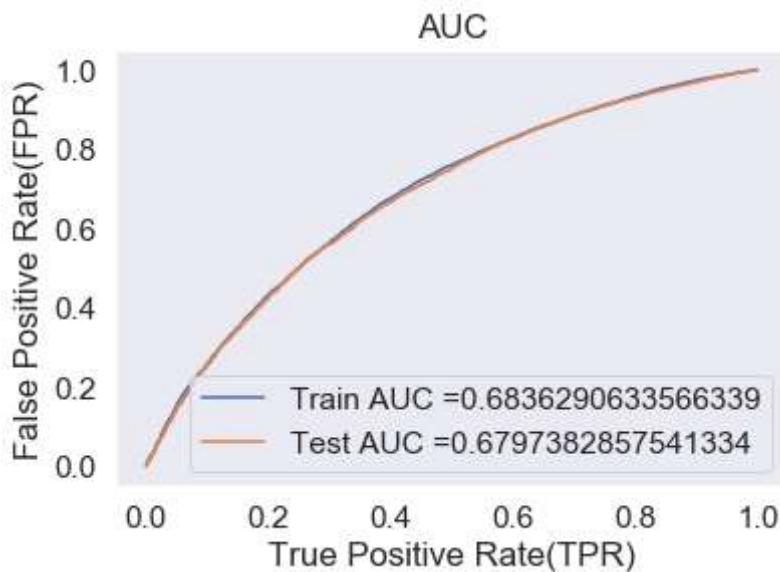
```
In [185]: 1 best_c_4 = 0.00195
```

In [186]:

```

1 from sklearn.metrics import roc_curve, auc
2
3 LGR_tfidf2v = LogisticRegression(C = best_c_4)
4
5 LGR_tfidf2v.fit(X_train4, y_train)
6 # roc_auc_score(y_true, y_score) the 2nd parameter should be probability est
7 # not the predicted outputs
8
9 y_train_pred = batch_predict(LGR_tfidf2v, X_train4)
10 y_test_pred = batch_predict(LGR_tfidf2v, X_test4)
11
12 train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
13 test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
14
15 plt.plot(train_fpr, train_tpr, label="Train AUC =" + str(auc(train_fpr, train_
16 plt.plot(test_fpr, test_tpr, label="Test AUC =" + str(auc(test_fpr, test_tpr))
17 plt.legend()
18 plt.xlabel("True Positive Rate(TPR)")
19 plt.ylabel("False Positive Rate(FPR)")
20 plt.title("AUC")
21 plt.grid()
22 plt.show()

```



## Train Confusion Matrix

In [187]:

```

1 from sklearn.metrics import confusion_matrix
2 import seaborn as sea
3 print("Train confusion matrix")
4 print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_f

```

```

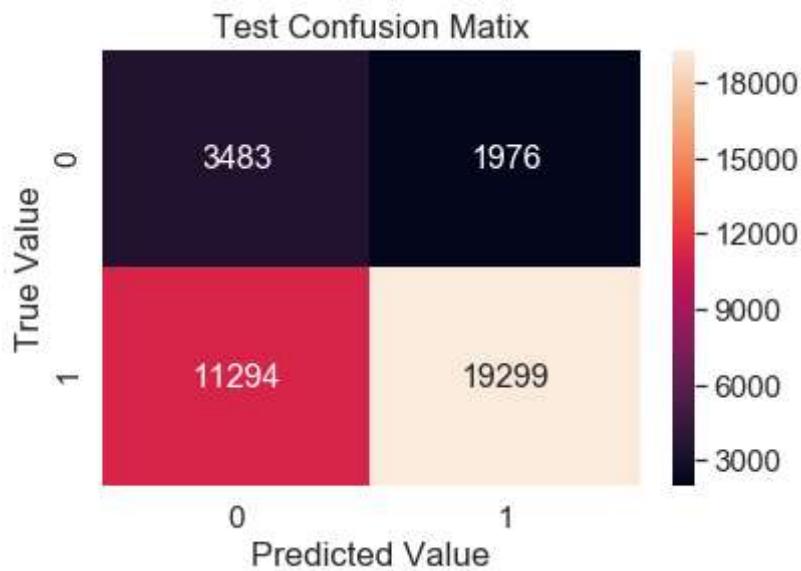
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.821
[[ 3713  3713]
 [10026 31589]]

```

```
In [188]: 1 train_confusion_matrix = pd.DataFrame(confusion_matrix(y_test,predict(y_test))
2
3 sea.set(font_scale=1.4)
4 sea.heatmap(train_confusion_matrix, annot = True, annot_kws={"size":16}, fmt
5 plt.xlabel("Predicted Value")
6 plt.ylabel("True Value")
7 plt.title("Test Confusion Matix")
```

the maximum value of tpr\*(1-fpr) 0.2499999244983697 for threshold 0.841

Out[188]: Text(0.5,1,'Test Confusion Matix')



## Test Confusion Matrix

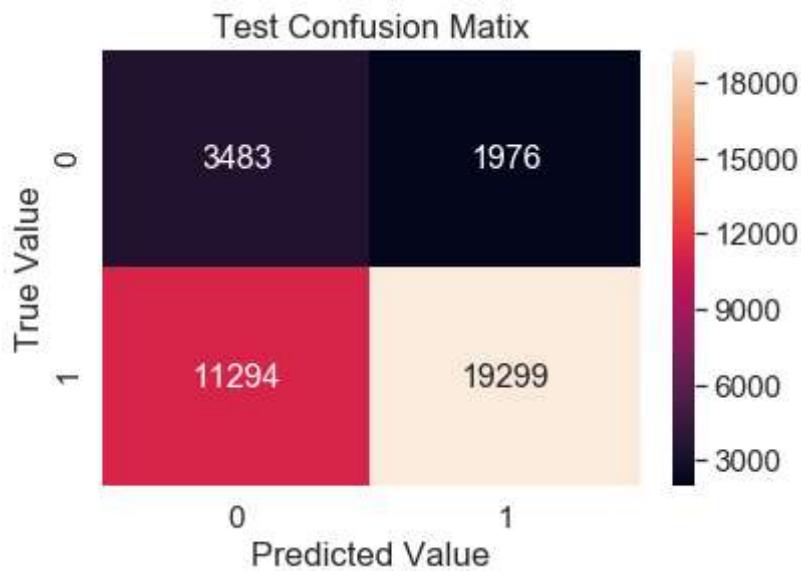
```
In [189]: 1 print("Test confusion matrix")
2 print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr,
```

Test confusion matrix  
the maximum value of tpr\*(1-fpr) 0.2499999244983697 for threshold 0.841  
[[ 3483 1976]  
 [11294 19299]]

```
In [190]: 1 train_confusion_matrix = pd.DataFrame(confusion_matrix(y_test,predict(y_test))
2
3 sea.set(font_scale=1.4)
4 sea.heatmap(train_confusion_matrix, annot = True, annot_kws={"size":16}, fmt
5 plt.xlabel("Predicted Value")
6 plt.ylabel("True Value")
7 plt.title("Test Confusion Matix")
```

the maximum value of tpr\*(1-fpr) 0.2499999244983697 for threshold 0.841

Out[190]: Text(0.5,1,'Test Confusion Matix')



### 3. Conclusion

```
In [193]: 1 # Please compare all your models using Prettytable Library
2 # http://zetcode.com/python/prettytable/
3
4 from prettytable import PrettyTable
5 TB = PrettyTable()
6 TB.field_names = ["Vectorizer", "C:Hyperparameter", "Train_AUC", "Test_Auc"]
7 TB.title = "Logistic Regression"
8 TB.add_row(["BOW", 0.0156, 0.79, 0.68])
9 TB.add_row(["TFIDF", 0.0039, 0.72, 0.66])
10 TB.add_row(["AvgW2v", 0.0039, 0.68, 0.67])
11 TB.add_row(["Tf-Idf", 0.00195, 0.68, 0.67])
12 print(TB)
```

Vectorizer	C:Hyperparameter	Train_AUC	Test_Auc
BOW	0.0156	0.79	0.68
TFIDF	0.0039	0.72	0.66
AvgW2v	0.0039	0.68	0.67
Tf-Idf	0.00195	0.68	0.67

## Observations:

From above we can see the BOW and TFIDF encoding contain the Project\_Essays and Project\_titles in those models. So, that we can say that Text Data also plays major role in predicting the output.