

# 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	Description
<code>project_id</code>	A unique identifier for the proposed project <b>Example:</b> p036502

Feature	Description
<b>project_title</b>	Title of the project. <b>Examples:</b> <ul style="list-style-type: none"> <li>• Art Will Make You Happy</li> <li>• First Grade Fun</li> </ul>
<b>project_grade_category</b>	Grade level of students for which the project is targeted. One of the follow enumerated values: <ul style="list-style-type: none"> <li>• Grades PreK-2</li> <li>• Grades 3-5</li> <li>• Grades 6-8</li> <li>• Grades 9-12</li> </ul>
<b>project_subject_categories</b>	One or more (comma-separated) su categories for the project from the fo enumerated list of values: <ul style="list-style-type: none"> <li>• Applied Learning</li> <li>• Care &amp; Hunger</li> <li>• Health &amp; Sports</li> <li>• History &amp; Civics</li> <li>• Literacy &amp; Language</li> <li>• Math &amp; Science</li> <li>• Music &amp; The Arts</li> <li>• Special Needs</li> <li>• Warmth</li> </ul> <b>Examples:</b> <ul style="list-style-type: none"> <li>• Music &amp; The Arts</li> <li>• Literacy &amp; Language, Ma &amp; Science</li> </ul>

Feature	Description
<b>school_state</b>	State where school is located ( <a href="#">Two-Digit U.S. postal code</a> ). <b>Example:</b> WY
<b>project_subject_subcategories</b>	One or more (comma-separated) subcategories for the project. <b>Example:</b> <ul style="list-style-type: none"> <li>• Literacy</li> <li>• Literature &amp; Writing, Social Sciences</li> </ul>
<b>project_resource_summary</b>	An explanation of the resources needed for the project. <b>Example:</b> <ul style="list-style-type: none"> <li>• My students need hands-on literacy materials to manage sensory needs!</li> </ul>
<b>project_essay_1</b>	First application essay*
<b>project_essay_2</b>	Second application essay*
<b>project_essay_3</b>	Third application essay*
<b>project_essay_4</b>	Fourth application essay*
<b>project_submitted_datetime</b>	Datetime when project application was submitted. <b>Example:</b> 2016-04-28 12:43:56.245
<b>teacher_id</b>	A unique identifier for the teacher of proposed project. <b>Example:</b> bdf8baa8fedef6bfeec7ae4ff1c

Feature	Description
<b>teacher_prefix</b>	Teacher's title. One of the following enumerated values: <ul style="list-style-type: none"> <li>• nan</li> <li>• Dr.</li> <li>• Mr.</li> <li>• Mrs.</li> <li>• Ms.</li> <li>• Teacher.</li> </ul>
<b>teacher_number_of_previously_posted_projects</b>	Number of project applications previously submitted by the same teacher. <b>Example:</b> 2

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

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

Feature	Description
<b>id</b>	A <code>project_id</code> value from the <code>train.csv</code> 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
project_is_approved	A binary flag indicating whether DonorsChoose approved the project. A value of 0 indicates the project was not approved, and a value of 1 indicates the project was approved.



## Notes on the Essay Data

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

- \_\_project\_essay\_1:\_\_ "Introduce us to your classroom"
- \_\_project\_essay\_2:\_\_ "Tell us more about your students"
- \_\_project\_essay\_3:\_\_ "Describe how your students will use the materials you're requesting"
- \_\_project\_essay\_3:\_\_ "Close by sharing why your project will make a difference"

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

- \_\_project\_essay\_1:\_\_ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- \_\_project\_essay\_2:\_\_ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project\_submitted\_datetime of 2016-05-17 and later, the values of project\_essay\_3 and project\_essay\_4 will be NaN.

```
In [88]: %matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
```

```

import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

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

```

IOPub data rate exceeded.  
 The notebook server will temporarily stop sending output  
 to the client in order to avoid crashing it.  
 To change this limit, set the config variable  
 `--NotebookApp.iopub\_data\_rate\_limit`.

## 1.1 Reading Data

```
In [89]: project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

```
In [90]: print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

Number of data points in train data (109248, 17)

-----

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

```
In [91]: print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

Number of data points in train data (1541272, 4)

['id' 'description' 'quantity' 'price']

Out[91]:

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

## 1.2 preprocessing of project\_subject\_categories

```
In [92]: categories = list(project_data['project_subject_categories'].values)
```

```

# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

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

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

```



## 1.3 preprocessing of project\_subject\_subcategories

```
In [93]: sub_categories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

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

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

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

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
```

```

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

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

```

## 1.3 Text preprocessing

```

In [94]: # merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) + \
    project_data["project_essay_2"].map(str) + \
    project_data["project_essay_3"].map(str) + \
    project_data["project_essay_4"].map(str)

```

```

In [95]: project_data.head(2)

```

Out[95]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL

In [96]: `#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V`

In [97]: `# printing some random reviews  
print(project_data['essay'].values[0])  
print("="*50)  
print(project_data['essay'].values[150])  
print("="*50)  
print(project_data['essay'].values[1000])  
print("="*50)  
print(project_data['essay'].values[20000])  
  
print("="*50)`

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.\n"\"The limits of your language are the limits of your world.\n"\"-Ludwig Wittgenstein Our English learner's have a strong support system at home that begs for more resources. Many times our parents are learning to read and speak English alongside of their children. Sometimes this creates barriers for parents to be ab

le 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\nnnannan

=====

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\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my stud

ents 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\n\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\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\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\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 in turn fine motor skills. \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  
=====

```
In [98]: # https://stackoverflow.com/a/47091490/4084039
import re

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

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

```
In [99]: sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism

m. 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 in turn fine motor skills. \r\nThey also want to learn through games, my kids do not want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the 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 [100]: # \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\r', ' ')
sent = sent.replace('\n', ' ')
sent = sent.replace('\t', ' ')
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

r 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 [101]: #remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

My kindergarten students have varied disabilities ranging from speech and language delays cognitive 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 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 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 [102]: # https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves',
            'you', "you're", "you've", \
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
            'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its',
            'itself', 'they', 'them', 'their', \
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
            'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
            'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or',
```



```
'because', 'as', 'until', 'while', 'of', \
    'at', 'by', 'for', 'with', 'about', 'against', 'between',
'into', 'through', 'during', 'before', 'after', \
    'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out',
'on', 'off', 'over', 'under', 'again', 'further', \
    'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',
'ow', 'all', 'any', 'both', 'each', 'few', 'more', \
    'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so',
'o', 'than', 'too', 'very', \
    's', 't', 'can', 'will', 'just', 'don', "don't", 'should',
"should've", 'now', 'd', 'll', 'm', 'o', 're', \
    've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't",
'didn', "didn't", 'doesn', "doesn't", 'hadn', \
    "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
'n't", 'ma', 'mightn', "mightn't", 'mustn', \
    "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn',
"shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
    'won', "won't", 'wouldn', "wouldn't"]
```

```
In [103]: # Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['essay'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [02:14<00:00, 812.41it/s]
```

```
In [104]: # after preprocessing
preprocessed_essays[20000]
```

```
Out[104]: 'my kindergarten students varied disabilities ranging speech language d
elays cognitive delays gross fine motor delays autism they eager beaver
s always strive work hardest working past limitations the materials one
s i seek students i teach title i school students receive free reduced
price lunch despite disabilities limitations students love coming schoo
l come eager learn explore have ever felt like ants pants needed groove
move meeting this kids feel time the want able move learn say wobble ch
airs answer i love develop core enhances gross motor turn fine motor sk
ills they also want learn games kids not want sit worksheets they want
learn count jumping playing physical engagement key success the number
toss color shape mats make happen my students forget work fun 6 year ol
d deserves nannan'
```

```
In [235]: project_data["preprocessed_essays"] = preprocessed_essays

essay_word_count = []

for sentence in project_data["preprocessed_essays"] :
    word = len(sentence.split())
    essay_word_count.append(word)

project_data["essay_word_count"] = essay_word_count

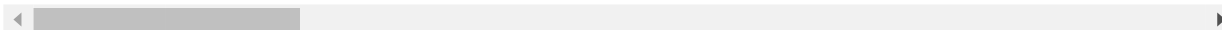
project_data.head(5)
```

Out[235]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
1	140945	p258326	897464ce9ddc600bcd1151f324dd63a	Mr.	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX

5 rows × 23 columns



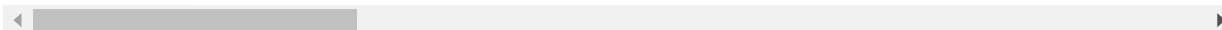
```
In [245]: sid = SentimentIntensityAnalyzer()
neg = []
pos = []
neu = []
```

```
100%|██████████████████████████████████████████████████████████████████████████|  
██████████ | 109248/109248 [25:12<00:00, 72.23it/s]
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX

5 rows × 27 columns



## 1.4 Preprocessing of `project\_title`

```
In [105]: # printing some project titles.
for i in range (0,21):

    print(project_data['project_title'].values[i])
    print("="*50)
```

```
Educational Support for English Learners at Home
=====
Wanted: Projector for Hungry Learners
=====
Soccer Equipment for AWESOME Middle School Students
=====
Techie Kindergarteners
```

```

=====
Interactive Math Tools
=====
Flexible Seating for Mrs. Jarvis' Terrific Third Graders!!
=====
Chromebooks for Special Education Reading Program
=====
It's the 21st Century
=====
Targeting More Success in Class
=====
Just For the Love of Reading--\r\nPure Pleasure
=====
Reading Changes Lives
=====
Elevating Academics and Parent Rapports Through Technology
=====
Building Life Science Experiences
=====
Everyone deserves to be heard!
=====
TABLETS CAN SHOW US THE WORLD
=====
Making Recess Active
=====
Making Great LEAP's With Leapfrog!
=====
Technology Teaches Tomorrow's Talents Today
=====
Test Time
=====
Wiggling Our Way to Success
=====
Magic Carpet Ride in Our Library
=====

```

```

In [106]: preprocessed_titles = []

for dataset in tqdm(project_data['project_title'].values):

```

```
data = decontracted(dataset) # Replacing some specific and general
short form into proper word/stopword.
data = re.sub(r"it's", "it is", data) # Replacing it's with it is a
s it is not part of function decontracted
data = data.replace('\\r', ' ') # Replacing \r with space
data = data.replace('\\\"', ' ') # Replacing \" with space
data = data.replace('\\n', ' ') # Replacing \n with space
data = re.sub('[^A-Za-z0-9]+', ' ', data) # Replacing special chara
cters with space
data = re.sub("\S*\d\S*", "", data).strip() # Trimming numbers cont
aining digits

data = ' '.join(e for e in data.split() if e not in stopwords) # Re
moving stopwords
preprocessed_titles.append(data.lower().strip()) # Creating array i
n all the lower cases.
```

```
In [107]: for i in range(0,21):
           print(preprocessed_titles[i])
           print("="*50)
```

```
targeting more success class
=====
just for love reading pure pleasure
=====
reading changes lives
=====
elevating academics parent rapports through technology
=====
building life science experiences
=====
everyone deserves heard
=====
tablets can show us the world
=====
making recess active
=====
making great leap with leapfrog
=====
technology teaches tomorrow talents today
=====
test time
=====
wiggling our way success
=====
magic carpet ride our library
=====
```

```
In [234]: project_data["preprocessed_titles"] = preprocessed_titles

title_word_count = []

for sentence in project_data["preprocessed_titles"] :
    word = len(sentence.split())
    title_word_count.append(word)

project_data["title_word_count"] = title_word_count

project_data.head(5)
```



Out[234]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bcd1151f324dd63a	Mr.	FL
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX

5 rows × 21 columns



## 1.5 Preparing data for models

In [108]: `project_data.columns`

Out[108]: 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 (optinal)
- quantity : numerical (optinal)
- teacher\_number\_of\_previously\_posted\_projects : numerical
- price : numerical

### 1.5.1 Vectorizing Categorical data

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

```
In [109]: # we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encoding ", categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encoding (109248, 9)
```

```
In [110]: # we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].values)
```

```
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.s
hape)
```

```
['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 matrix after one hot encodig (109248, 30)
```

```
In [111]: school_state_vectorizer = CountVectorizer(lowercase=False, binary=True)
school_state_vectorizer.fit(project_data['school_state'].values)
print(school_state_vectorizer.get_feature_names())
```

```
school_state_one_hot = school_state_vectorizer.transform(project_data['school_state'].values)
print("Shape of matrix after one hot encodig ",school_state_one_hot.shape)
print("the type of count vectorizer ",type(school_state_one_hot))
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV', 'WY']
```

```
Shape of matrix after one hot encodig (109248, 51)
the type of count vectorizer <class 'scipy.sparse.csr.csr_matrix'>
```

```
In [112]: # https://www.geeksforgeeks.org/python-pandas-dataframe-fillna-to-replace-null-values-in-dataframe/
project_data["teacher_prefix"].fillna("No_Prefix", inplace = True)
```

```
teacher_prefix_vectorizer = CountVectorizer(lowercase=False, binary=True)
teacher_prefix_vectorizer.fit(project_data['teacher_prefix'].values)
```

```
print(teacher_prefix_vectorizer.get_feature_names())

teacher_prefix_one_hot = teacher_prefix_vectorizer.transform(project_data['teacher_prefix'].values)
print("Shape of matrix after one hot encoding ", teacher_prefix_one_hot.shape)
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'No_Prefix', 'Teacher']
Shape of matrix after one hot encoding (109248, 6)
```

```
In [113]: my_grade_counter = Counter()

for project_grade in project_data['project_grade_category'].values:
    if (' ' in project_grade):
        project_grade = project_grade.replace(" ", "~")

    my_grade_counter.update(project_grade.split())

project_grade_cat_dict = dict(my_grade_counter)
sorted_project_grade_cat_dict = dict(sorted(project_grade_cat_dict.items(), key=lambda kv: kv[1]))

grade_cat_vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_cat_dict.keys()), lowercase=False, binary=True)
grade_cat_vectorizer.fit(project_data['project_grade_category'].values)
print(grade_cat_vectorizer.get_feature_names())

grade_cat_one_hot = grade_cat_vectorizer.transform(project_data['project_grade_category'].values)
print("Shape of matrix after one hot encoding ", grade_cat_one_hot.shape)

['Grades~9-12', 'Grades~6-8', 'Grades~3-5', 'Grades~PreK-2']
Shape of matrix after one hot encoding (109248, 4)
```

## 1.5.2 Vectorizing Text data

### 1.5.2.1 Bag of words

```
In [114]: # We are considering only the words which appeared in at least 10 documents (rows or projects).
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encoding ", text_bow.shape)
```

Shape of matrix after one hot encoding (109248, 16623)

```
In [115]: titles_vectorizer = CountVectorizer(min_df=10)
titles_bow = titles_vectorizer.fit_transform(preprocessed_titles)
print("some sample features(unique words in the corpus)", titles_vectorizer.get_feature_names()[0:10])
print("Shape of matrix after one hot encoding ", titles_bow.shape)
print("the type of count vectorizer ", type(titles_bow))
print("the number of unique words ", titles_bow.get_shape()[1])
```

some sample features(unique words in the corpus) ['abc', 'abcs', 'abilities', 'ability', 'able', 'aboard', 'about', 'above', 'abstract', 'academic']

Shape of matrix after one hot encoding (109248, 3290)

the type of count vectorizer <class 'scipy.sparse.csr.csr\_matrix'>

the number of unique words 3290

### 1.5.2.2 TFIDF vectorizer

```
In [116]: from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encoding ", text_tfidf.shape)
```

Shape of matrix after one hot encoding (109248, 16623)

### 1.5.2.3 Using Pretrained Models: Avg W2V

```
In [117]: '''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# =====
Output:

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

# =====

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

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

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and o
```

```

ur coupus", \
    len(inter_words), "(" , np.round(len(inter_words)/len(words)*100,
3), "%)")

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

# stronging variables into pickle files python: http://www.jessicayung.
com/how-to-use-pickle-to-save-and-load-variables-in-python/

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

'''

```

```

Out[117]: '\n# Reading glove vectors in python: https://stackoverflow.com/a/38230
349/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove
Model")\n    f = open(gloveFile,\r', encoding="utf8")\n    model = {}
\n    for line in tqdm(f):\n        splitLine = line.split()\n        w
ord = splitLine[0]\n        embedding = np.array([float(val) for val in
splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",le
n(model)," words loaded!")\n    return model\nmodel = loadGloveModel
(\glove.42B.300d.txt')\n\n# =====\nOutput:\n
\nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495
words loaded!\n\n# =====\n\nwords = []\nfor i in
preproced_texts:\n    words.extend(i.split(\ ' '))\n\nfor i in preproce
d_titles:\n    words.extend(i.split(\ ' '))\nprint("all the words in th
e coupus", len(words))\nwords = set(words)\nprint("the unique words in
the coupus", len(words))\n\ninter_words = set(model.keys()).intersectio
n(words)\nprint("The number of words that are present in both glove vec
tors and our coupus", len(inter_words), "(" , np.round(len(inter_wor
ds)/len(words)*100,3), "%)")\n\nwords_courpus = {}\nwords_glove = set(mo
del.keys())\nfor i in words:\n    if i in words_glove:\n        words_c
'''

```



```

ourpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n
\n\n# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pickle\nwith open('glove_vectors', 'wb') as f:\n    pickle.dump(words_courpus, f)\n\n\n
```

```

In [118]: # stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words = set(model.keys())

```

```

In [119]: # average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))

```

```

100%|████████████████████████████████████████████████████████████████████████████████| 109248/109248 [01:02<00:00, 1736.72it/s]

```

```

109248
300

```

### 1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

```
In [120]: # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

```
In [121]: # average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|███████████████████████████████████████████████████████████|
██████████████████████████████████████████████████████████████ | 109248/109248 [08:18<00:00, 219.13it/s]
```

109248  
300

```

In [122]: # TFIDF on project titles
titles_tfidf_vectorizer = TfidfVectorizer(min_df=10)
titles_tfidf = titles_tfidf_vectorizer.fit_transform(preprocessed_titles)
print("some sample features(unique words in the corpus)",titles_tfidf_vectorizer.get_feature_names()[10:21])
print("Shape of matrix after one hot encoding ",titles_tfidf.shape)

```

some sample features(unique words in the corpus) ['academics', 'academy', 'acceptance', 'access', 'accessibility', 'accessible', 'accessing', 'accessories', 'ace', 'achieve', 'achievement']  
 Shape of matrix after one hot encoding (109248, 3290)

```

In [123]: # AVG W2V on project title
avg_w2v_titles_vectors = [];

for sentence in tqdm(preprocessed_titles):

    vector_titles = np.zeros(300)
    cnt_words_titles = 0;

    for word in sentence.split():

        if word in glove_words:

            vector += model[word]
            cnt_words_titles += 1

    if cnt_words_titles != 0:

        vector_titles /= cnt_words_titles

    avg_w2v_titles_vectors.append(vector_titles)

print(len(avg_w2v_titles_vectors))
print(len(avg_w2v_titles_vectors[0]))

```

100%|

██████ | 109248/109248 [00:02<00:00, 40270.97it/s]

109248  
300

```
In [124]: # TFIDF weighted W2V on project_title
titles_tfidf_model = TfidfVectorizer()
titles_tfidf_model.fit(preprocessed_titles)
titles_dictionary = dict(zip(titles_tfidf_model.get_feature_names(), list(titles_tfidf_model.idf_)))
titles_tfidf_words = set(titles_tfidf_model.get_feature_names())
```

```
In [125]: titles_tfidf_w2v_vectors = [];

for titles_sentence in tqdm(preprocessed_titles):

    titles_vector = np.zeros(300)
    titles_tfidf_weight = 0;

    for word in titles_sentence.split():

        if (word in glove_words) and (word in titles_tfidf_words):

            titles_vec = model[word]

            titles_tf_idf = titles_dictionary[word]*(titles_sentence.count(word)/len(titles_sentence.split()))
            titles_vector += (titles_vec * titles_tf_idf)
            titles_tfidf_weight += titles_tf_idf

    if titles_tfidf_weight != 0:

        titles_vector /= titles_tfidf_weight

    titles_tfidf_w2v_vectors.append(titles_vector)

print(len(titles_tfidf_w2v_vectors))
print(len(titles_tfidf_w2v_vectors[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 109248/109248 [00:06<00:00, 15934.72it/s]
```

```
109248
300
```

### 1.5.3 Vectorizing Numerical features

```
In [126]: price_data = resource_data.groupby('id').agg({'price': 'sum', 'quantity': 'sum'}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

```
In [127]: # check this one: https://www.youtube.com/watch?v=0H0q0cIn3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScaler.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 21
3.03 329. ... 399. 287.73 5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding
the mean and standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(p
rice_scalar.var_[0])}")

# Now standardize the data with above mean and variance.
price_standardized = price_scalar.transform(project_data['price'].value
s.reshape(-1, 1))
```

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
```

```
In [128]: price_standardized
```

```
Out[128]: array([[ -0.3905327 ],
```

```
[ 0.00239637],  
[ 0.59519138],  
...  
[-0.15825829],  
[-0.61243967],  
[-0.51216657]])
```

### 1.5.4 Merging all the above features

- we need to merge all the numerical vectors i.e categorical, text, numerical vectors

```
In [129]: print(categories_one_hot.shape)  
print(sub_categories_one_hot.shape)  
print(text_bow.shape)  
print(price_standardized.shape)
```

```
(109248, 9)  
(109248, 30)  
(109248, 16623)  
(109248, 1)
```

```
In [130]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039  
from scipy.sparse import hstack  
# with the same hstack function we are concatenating a sparse matrix and a dense matrix :)  
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))  
X.shape
```

```
Out[130]: (109248, 16663)
```

### Computing Sentiment Scores

```
In [131]: import nltk  
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```

# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i te
ach the smallest students with the biggest enthusiasm \
for learning my students learn in many different ways using all of our
senses and multiple intelligences i use a wide range\
of techniques to help all my students succeed students in my class come
from a variety of different backgrounds which makes\
for wonderful sharing of experiences and cultures including native amer
icans our school is a caring community of successful \
learners which can be seen through collaborative student project based
learning in and out of the classroom kindergarteners \
in my class love to work with hands on materials and have many differen
t opportunities to practice a skill before it is\
mastered having the social skills to work cooperatively with friends is
a crucial aspect of the kindergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition m
y students love to role play in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we
try cooking with real food i will take their idea \
and create common core cooking lessons where we learn important math an
d writing concepts while cooking delicious healthy \
food for snack time my students will have a grounded appreciation for t
he work that went into making the food and knowledge \
of where the ingredients came from as well as how it is healthy for the
ir bodies this project would expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own ap
ples to make homemade applesauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we wi
ll also create our own cookbooks to be printed and \
shared with families students will gain math and literature skills as w
ell as a life long enjoyment for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

```

```
for k in ss:
    print('{0}: {1}'.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# 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 paramter tuning (find best hyper parameters corresponding the algorithm that you choose)

- Find the best hyper parameter which will give the maximum [AUC](#) value
- Find the best hyper paramter using k-fold cross validation or simple cross validation data
- Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning



### 3. 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](#) with predicted and original labels of test data points. Please visualize your confusion matrices using [seaborn heatmaps](#).



### 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 paramter 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 \[prettytable library link\]\(#\)](#)



### Note: Data Leakage

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

## 2. Logistic Regression

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

```
In [132]: approved_project = project_data['project_is_approved'].values  
project_data.drop(['project_is_approved'], axis=1, inplace=True)  
project_data.head(1)
```

Out[132]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	p
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2

```
In [246]: # Data splitting

from sklearn.model_selection import train_test_split

# Splitting in train and test
X_train, X_test, y_train, y_test = train_test_split(project_data, approved_project, test_size=0.33, stratify=approved_project)

# Splitting in Train Test and Cross Validation
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

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

```
In [134]: # Vectorizing Categories on Train, Test and CV data
from sklearn.feature_extraction.text import CountVectorizer

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

# Fit only to train data
vectorizer.fit(X_train['clean_categories'].values)

# Transform to train, test and CV data
X_Train_categories_one_hot = vectorizer.transform(X_train['clean_categories'].values)
X_Test_categories_one_hot = vectorizer.transform(X_test['clean_categories'].values)
X_CV_categories_one_hot = vectorizer.transform(X_cv['clean_categories'].values)

print("Shape of train matrix after one hot encoding ", X_Train_categories_one_hot.shape)
print("Shape of test matrix after one hot encoding ", X_Test_categories_one_hot.shape)
```

```
print("Shape of cv matrix after one hot encodig ",X_CV_categories_one_hot.shape)
```

Shape of train matrix after one hot encodig (49041, 9)

Shape of test matrix after one hot encodig (36052, 9)

Shape of cv matrix after one hot encodig (24155, 9)

In [135]: *# Vectorizing subcategories on train, test and cv*

```
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
```

```
vectorizer.fit(X_train['clean_subcategories'].values)
```

```
X_Train_sub_categories_one_hot = vectorizer.transform(X_train['clean_subcategories'].values)
```

```
X_Test_sub_categories_one_hot = vectorizer.transform(X_test['clean_subcategories'].values)
```

```
X_CV_sub_categories_one_hot = vectorizer.transform(X_cv['clean_subcategories'].values)
```

```
print("Shape of train matrix after one hot encodig ",X_Train_sub_categories_one_hot.shape)
```

```
print("Shape of test matrix after one hot encodig ",X_Test_sub_categories_one_hot.shape)
```

```
print("Shape of cv matrix after one hot encodig ",X_CV_sub_categories_one_hot.shape)
```

Shape of train matrix after one 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 [136]: *# Vectorizing school state on train , test and cv*

```
school_state_vectorizer = CountVectorizer(lowercase=False, binary=True)
```

```
school_state_vectorizer.fit(X_train['school_state'].values)
```

```
print(school_state_vectorizer.get_feature_names())
```

```
X_Train_school_state_one_hot = school_state_vectorizer.transform(X_train['school_state'].values)
```

```

n['school_state'].values)
X_Test_school_state_one_hot = school_state_vectorizer.transform(X_test[
'school_state'].values)
X_CV_school_state_one_hot = school_state_vectorizer.transform(X_cv['sch
ool_state'].values)

print("Shape of train matrix after one hot encodig ",X_Train_school_sta
te_one_hot.shape)
print("Shape of test matrix after one hot encodig ",X_Test_school_state
_one_hot.shape)
print("Shape of cv matrix after one hot encodig ",X_CV_school_state_one
_hot.shape)

print("the type of count vectorizer ",type(X_Train_school_state_one_hot
))
print("the type of count vectorizer ",type(X_Test_school_state_one_hot
))
print("the type of count vectorizer ",type(X_CV_school_state_one_hot))

['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'H
I', 'IA', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI',
'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY',
'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT',
'WA', 'WI', 'WV', 'WY']
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)
the type of count vectorizer <class 'scipy.sparse.csr.csr_matrix'>
the type of count vectorizer <class 'scipy.sparse.csr.csr_matrix'>
the type of count vectorizer <class 'scipy.sparse.csr.csr_matrix'>

```

```

In [137]: # Vectorizing teacher prefix on train , test and cv

project_data["teacher_prefix"].fillna("No_Prefix", inplace = True)

teacher_prefix_vectorizer = CountVectorizer(lowercase=False, binary=True)
teacher_prefix_vectorizer.fit(X_train['teacher_prefix'].values)

```

```

print(teacher_prefix_vectorizer.get_feature_names())

X_Train_teacher_prefix_one_hot = teacher_prefix_vectorizer.transform(X_train['teacher_prefix'].values)
X_Test_teacher_prefix_one_hot = teacher_prefix_vectorizer.transform(X_test['teacher_prefix'].values)
X_CV_teacher_prefix_one_hot = teacher_prefix_vectorizer.transform(X_cv['teacher_prefix'].values)

print("Shape of train matrix after one hot encoding ", X_Train_teacher_prefix_one_hot.shape)
print("Shape of test matrix after one hot encoding ", X_Test_teacher_prefix_one_hot.shape)
print("Shape of cv matrix after one hot encoding ", X_CV_teacher_prefix_one_hot.shape)

['Dr', 'Mr', 'Mrs', 'Ms', 'No_Prefix', 'Teacher']
Shape of train matrix after one hot encoding (49041, 6)
Shape of test matrix after one hot encoding (36052, 6)
Shape of cv matrix after one hot encoding (24155, 6)

```

```

In [138]: # Vectorizing grade category on train , test and cv

my_grade_counter = Counter()

for project_grade in project_data['project_grade_category'].values:
    if (' ' in project_grade):
        project_grade = project_grade.replace(" ", "~")

    my_grade_counter.update(project_grade.split())

project_grade_cat_dict = dict(my_grade_counter)
sorted_project_grade_cat_dict = dict(sorted(project_grade_cat_dict.items(), key=lambda kv: kv[1]))

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

```

```

grade_cat_vectorizer.fit(X_train['project_grade_category'].values)
print(grade_cat_vectorizer.get_feature_names())

X_Train_grade_cat_one_hot = grade_cat_vectorizer.transform(X_train['project_grade_category'].values)
X_Test_grade_cat_one_hot = grade_cat_vectorizer.transform(X_test['project_grade_category'].values)
X_CV_grade_cat_one_hot = grade_cat_vectorizer.transform(X_cv['project_grade_category'].values)

print("Shape of train matrix after one hot encoding ", X_Train_grade_cat_one_hot.shape)
print("Shape of test matrix after one hot encoding ", X_Test_grade_cat_one_hot.shape)
print("Shape of cv matrix after one hot encoding ", X_CV_grade_cat_one_hot.shape)

['Grades~9-12', 'Grades~6-8', 'Grades~3-5', 'Grades~PreK-2']
Shape of train matrix after one hot encoding (49041, 4)
Shape of test matrix after one hot encoding (36052, 4)
Shape of cv matrix after one hot encoding (24155, 4)

```

## 2.3 Make Data Model Ready: encoding essay, and project\_title

```

In [139]: # merge two column text dataframe:
X_train["essay"] = X_train["project_essay_1"].map(str) + \
                  X_train["project_essay_2"].map(str) + \
                  X_train["project_essay_3"].map(str) + \
                  X_train["project_essay_4"].map(str)

```

```

In [140]: # preprocessing essay train data
from tqdm import tqdm
X_Train_preprocessed_essays = []
# tqdm is for printing the status bar
for sentence in tqdm(X_train['essay'].values):
    X_Train_essay_sent = decontracted(sentence)

```







```

essay_sent_pos_train = normalizer.transform(X_train['pos'].values.reshape(-1,1))
essay_sent_pos_cv = normalizer.transform(X_cv['pos'].values.reshape(-1,1))
essay_sent_pos_test = normalizer.transform(X_test['pos'].values.reshape(-1,1))

print("After vectorizations")
print(essay_sent_pos_train.shape, y_train.shape)
print(essay_sent_pos_cv.shape, y_cv.shape)
print(essay_sent_pos_test.shape, y_test.shape)
print("="*100)

normalizer.fit(X_train['neg'].values.reshape(-1,1))

essay_sent_neg_train = normalizer.transform(X_train['neg'].values.reshape(-1,1))
essay_sent_neg_cv = normalizer.transform(X_cv['neg'].values.reshape(-1,1))
essay_sent_neg_test = normalizer.transform(X_test['neg'].values.reshape(-1,1))

print("After vectorizations")
print(essay_sent_neg_train.shape, y_train.shape)
print(essay_sent_neg_cv.shape, y_cv.shape)
print(essay_sent_neg_test.shape, y_test.shape)
print("="*100)

normalizer.fit(X_train['neu'].values.reshape(-1,1))

essay_sent_neu_train = normalizer.transform(X_train['neu'].values.reshape(-1,1))
essay_sent_neu_cv = normalizer.transform(X_cv['neu'].values.reshape(-1,1))
essay_sent_neu_test = normalizer.transform(X_test['neu'].values.reshape(-1,1))

print("After vectorizations")

```

```

print(essay_sent_neu_train.shape, y_train.shape)
print(essay_sent_neu_cv.shape, y_cv.shape)
print(essay_sent_neu_test.shape, y_test.shape)
print("="*100)

normalizer.fit(X_train['compound'].values.reshape(-1,1))

essay_sent_comp_train = normalizer.transform(X_train['compound'].values
.reshape(-1,1))
essay_sent_comp_cv = normalizer.transform(X_cv['compound'].values.reshape(-1,1))
essay_sent_comp_test = normalizer.transform(X_test['compound'].values.reshape(-1,1))

print("After vectorizations")
print(essay_sent_comp_train.shape, y_train.shape)
print(essay_sent_comp_cv.shape, y_cv.shape)
print(essay_sent_comp_test.shape, y_test.shape)
print("="*100)

```

```

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

```

```

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

```

```

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

```

```

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

```

=====

=====

```
# preprocessing project title test data
X_Test_preprocessed_titles = []

for dataset in tqdm(X_test['project_title'].values):
    data = decontracted(dataset) # Replacing some specific and general
    # short form into proper word/stopword.
    data = re.sub(r"it's", "it is", data) # Replacing it's with it is a
    # s it is not part of function decontracted
    data = data.replace('\r', ' ') # Replacing \r with space
    data = data.replace('\n', ' ') # Replacing \ with space
```



```

normalizer.fit(X_train['title_word_count'].values.reshape(-1,1))

title_word_count_train = normalizer.transform(X_train['title_word_coun
t'].values.reshape(-1,1))
title_word_count_cv = normalizer.transform(X_cv['title_word_count'].val
ues.reshape(-1,1))
title_word_count_test = normalizer.transform(X_test['title_word_count']
.values.reshape(-1,1))

print("After vectorizations")
print(title_word_count_train.shape, y_train.shape)
print(title_word_count_cv.shape, y_cv.shape)
print(title_word_count_test.shape, y_test.shape)
print("="*100)

```

```

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

```

```

In [146]: # BOW Essay train, test and cv data
# We are considering only the words which appeared in at least 10 docum
ents(rows or projects).
vectorizer = CountVectorizer(ngram_range=(2,2), min_df=10, max_features
= 5000)
vectorizer.fit(X_Train_preprocessed_essays)

X_Train_essay_bow = vectorizer.transform(X_Train_preprocessed_essays)
X_Test_essay_bow = vectorizer.transform(X_Test_preprocessed_essays)
X_CV_essay_bow = vectorizer.transform(X_CV_preprocessed_essays)

print("Shape of train matrix after one hot encodig ",X_Train_essay_bow.
shape)
print("Shape of test matrix after one hot encodig ",X_Test_essay_bow.sh
ape)
print("Shape of CV matrix after one hot encodig ",X_CV_essay_bow.shape)

```

```
Shape of train matrix after one hot encodig (49041, 147)
Shape of test matrix after one hot encodig (36052, 147)
Shape of CV matrix after one hot encodig (24155, 147)
```

```
In [147]: # BOW title train,test and cv data

titles_vectorizer = CountVectorizer(ngram_range=(2,2), min_df=10, max_f
eatures = 5000)
titles_vectorizer.fit(X_Train_preprocessed_titles)

X_Train_titles_bow = titles_vectorizer.transform(X_Train_preprocessed_t
itles)
X_Test_titles_bow = titles_vectorizer.transform(X_Test_preprocessed_tit
les)
X_CV_titles_bow = titles_vectorizer.transform(X_CV_preprocessed_titles)

print("some sample features(unique words in the corpus)",titles_vectori
zer.get_feature_names()[0:10])
print("Shape of train matrix after one hot encodig ",X_Train_titles_bow
.shape)
print("Shape of test matrix after one hot encodig ",X_Test_titles_bow.s
hape)
print("Shape of CV matrix after one hot encodig ",X_CV_titles_bow.shape
)

some sample features(unique words in the corpus) ['about it', 'about ou
r', 'about reading', 'about science', 'about that', 'about the', 'about
world', 'academic success', 'access technology', 'active bodies']
Shape of train matrix after one hot encodig (49041, 1659)
Shape of test matrix after one hot encodig (36052, 1659)
Shape of CV matrix after one hot encodig (24155, 1659)
```

```
In [148]: #TFIDF essay train,test and cv data

vectorizer = TfidfVectorizer(ngram_range=(2,2), min_df=10, max_features
= 5000)
vectorizer.fit(X_Train_preprocessed_essays)

X_Train_essay_tfidf = vectorizer.transform(X_Train_preprocessed_essays)
```

```
X_Test_essay_tfidf = vectorizer.transform(X_Test_preprocessed_essays)
X_CV_essay_tfidf = vectorizer.transform(X_CV_preprocessed_essays)

print("Shape of train matrix after one hot encoding ",X_Train_essay_tfidf.shape)
print("Shape of test matrix after one hot encoding ",X_Test_essay_tfidf.shape)
print("Shape of CV matrix after one hot encoding ",X_CV_essay_tfidf.shape)
```

```
Shape of train matrix after one hot encoding (49041, 147)
Shape of test matrix after one hot encoding (36052, 147)
Shape of CV matrix after one hot encoding (24155, 147)
```

```
In [149]: # TFIDF on project titles train,test and cv data

titles_tfidf_vectorizer = TfidfVectorizer(ngram_range=(2,2), min_df=10,
max_features = 5000)
titles_tfidf_vectorizer.fit(X_Train_preprocessed_titles)

X_Train_titles_tfidf = titles_vectorizer.transform(X_Train_preprocessed_titles)
X_Test_titles_tfidf = titles_vectorizer.transform(X_Test_preprocessed_titles)
X_CV_titles_tfidf = titles_vectorizer.transform(X_CV_preprocessed_titles)

print("Shape of train matrix after one hot encoding ",X_Train_titles_tfidf.shape)
print("Shape of test matrix after one hot encoding ",X_Test_titles_tfidf.shape)
print("Shape of CV matrix after one hot encoding ",X_CV_titles_tfidf.shape)
```

```
Shape of train matrix after one hot encoding (49041, 1659)
Shape of test matrix after one hot encoding (36052, 1659)
Shape of CV matrix after one hot encoding (24155, 1659)
```

```
In [150]: # average Word2Vec essay on train
```



```
100%|██████████████████████████████████████████████████████████████████████████████|  
██████████ | 49041/49041 [00:25<00:00, 1933.71it/s]
```

```
# average Word2Vec essay on test
# compute average word2vec for each review.
X_Test_avg_w2v_vectors = []; # the avg-w2v for each sentence/review is
    stored in this list
for sentence in tqdm(X_Test_preprocessed_essays): # for each review/sen
    tence
        vector = np.zeros(300) # as word vectors are of zero length
        cnt_words = 0; # num of words with a valid vector in the sentence/re
        view
        for word in sentence.split(): # for each word in a review/sentence
            if word in glove_words:
                vector += model[word]
                cnt_words += 1
        if cnt_words != 0:
            vector /= cnt_words
```

36052  
300

24155  
300

```
100%|██████████| 49041/49041 [00:01<00:00, 33126.4lit/s]
```

```
In [154]: # AVG W2V on project title test
X Test avg w2v titles vectors = [];
```

PDFCROWD

```
100%|██████████████████████████████████████████████████████████████████████████████|
██████████ | 36052/36052 [00:01<00:00, 32520.73it/s]
36052
300
```

```
print(len(X_CV_avg_w2v_titles_vectors))
print(len(X_CV_avg_w2v_titles_vectors[0]))
```

24155  
300

```
In [157]: # TFIDF w2v essay train
# compute average word2vec for each review.
X_Train_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review
# is stored in this list
for sentence in tqdm(X_Train_preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight = 0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split()))
            # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    X_Train_tfidf_w2v_vectors.append(vector)
```

```
100%|███████████████████████████████████████████████████████████|
██████████ | 49041/49041 [03:31<00:00, 232.11it/s]
```

```
In [158]: # TFIDF w2v essay test
# compute average word2vec for each review.
X_Test_tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review i
s stored in this list
for sentence in tqdm(X_Test_preprocessed_essays): # for each review/sen
tence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight = 0; # num of words with a valid vector in the sentenc
e/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and t
he tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentenc
e.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
        if tf_idf_weight != 0:
            vector /= tf_idf_weight
    X_Test_tfidf_w2v_vectors.append(vector)

print(len(X_Test_tfidf_w2v_vectors))
print(len(X_Test_tfidf_w2v_vectors[0]))
```

```
100% | ██████████
    | 36052/36052 [00:13<00:00, 2682.08it/s]
```

PDFCROWD











```

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

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

```

```

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

```

## 2.4 Applying Logistic Regression on different kind of featurization as mentioned in the instructions

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

**Set 1: categorical, numerical features + project\_title(BOW) + preprocessed\_eassay (BOW with bi-grams with min\_df=10 and max\_features=5000)**

```

In [169]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

# Train data stack

```

```

X_tr = hstack((X_Train_categories_one_hot,X_Train_sub_categories_one_hot,
X_Train_school_state_one_hot,
               X_Train_teacher_prefix_one_hot,X_Train_grade_cat_one_hot
,X_Train_essay_bow,X_Train_titles_bow,
               X_train_price_norm)).tocsr()

# CV data Stack
X_cr = hstack((X_CV_categories_one_hot,X_CV_sub_categories_one_hot,X_CV
_school_state_one_hot,
               X_CV_teacher_prefix_one_hot,X_CV_grade_cat_one_hot,X_CV_
essay_bow,X_CV_titles_bow,
               X_cv_price_norm)).tocsr()

# Test Data Stack
X_te = hstack((X_Test_categories_one_hot,X_Test_sub_categories_one_hot,
X_Test_school_state_one_hot,
               X_Test_teacher_prefix_one_hot,X_Test_grade_cat_one_hot,X
_Test_essay_bow,X_Test_titles_bow,
               X_test_price_norm)).tocsr()

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

```

```

Final Data matrix
(49041, 1907) (49041,)
(24155, 1907) (24155,)
(36052, 1907) (36052,)

```

```

=====
=====

```

```

In [170]: def batch_predict(clf, data):

           y_data_pred = []

           # Changing the shape of predicted data in the multiple of 1000
           tr_loop = data.shape[0] - data.shape[0]%1000

```

```

# Running the loop for each 1000th data
for i in range(0, tr_loop, 1000):

    # Predicting probability
    y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])

y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

return y_data_pred

```

```

In [174]: # https://scikit-learn.org/stable/modules/generated/sklearn.model\_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV

neigh = LogisticRegression()
parameters = {'C':[0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]}
clf = GridSearchCV(neigh, parameters, cv=10, scoring='roc_auc')
clf.fit(X_tr, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

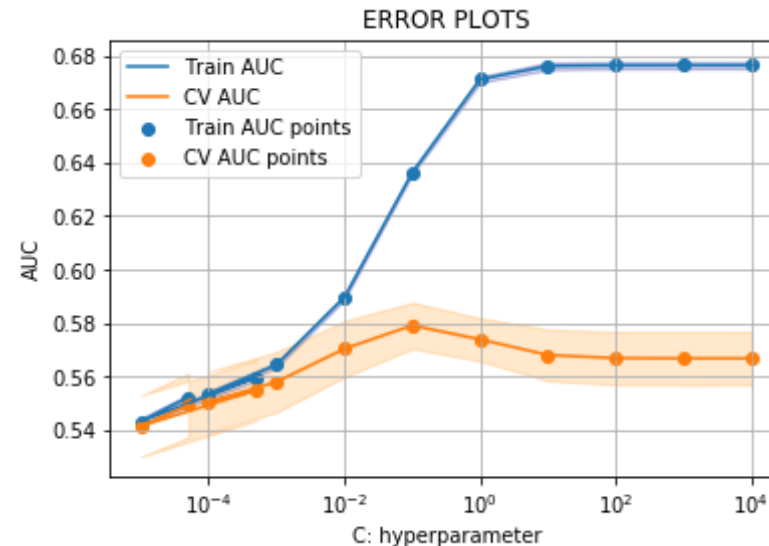
plt.plot(parameters['C'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

plt.scatter(parameters['C'], train_auc, label='Train AUC points')

```

```
plt.scatter(parameters['C'], cv_auc, label='CV AUC points')

plt.xscale("log")
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



```
In [189]: # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_
           _curve.html#sklearn.metrics.roc_curve
           from sklearn.metrics import roc_curve, auc

           neigh = LogisticRegression(C=0.5)
           neigh.fit(X_tr, y_train)
           # roc_auc_score(y_true, y_score) the 2nd parameter should be probabilit
           y estimates of the positive class
           # not the predicted outputs

           y_train_pred = batch_predict(neigh, X_tr)
```

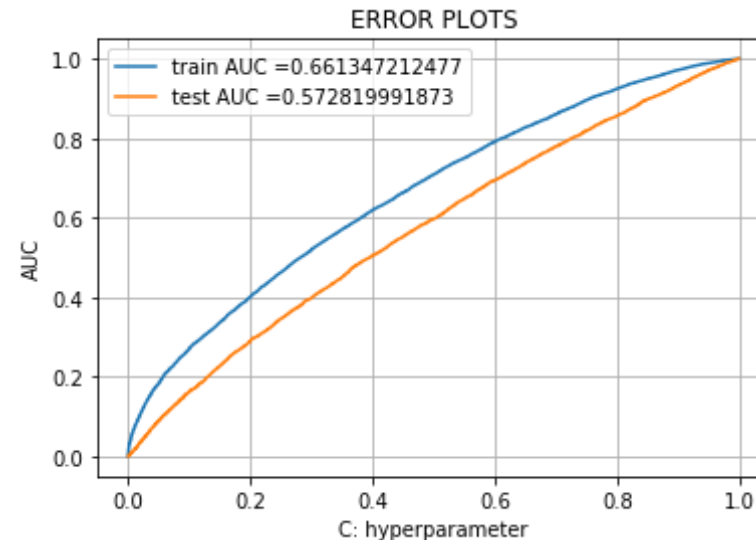
```

y_test_pred = batch_predict(neigh, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC =" + str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" + str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()

```



```

In [190]: # we are writing our own function for predict, with defined threshold
# we will pick a threshold that will give the least fpr
def predict(proba, threshold, fpr, tpr):

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

```

```

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

print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for th
reshold", np.round(t,3))
predictions = []
for i in proba:
    if i>=t:
        predictions.append(1)
    else:
        predictions.append(0)
return predictions

```

```

In [191]: print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, tr
ain_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test
_fpr, test_fpr)))

```

```

=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.249999836795 for threshold 0.833
[[ 3710  3716]
 [12084 29531]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.249999991611 for threshold 0.847
[[ 5438    21]
 [30415   178]]

```

```

In [192]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html
# https://stackoverflow.com/questions/29647749/seaborn-showing-scientif
ic-notation-in-heatmap-for-3-digit-numbers

# Train Confusion Matrix Heatmap
train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred

```

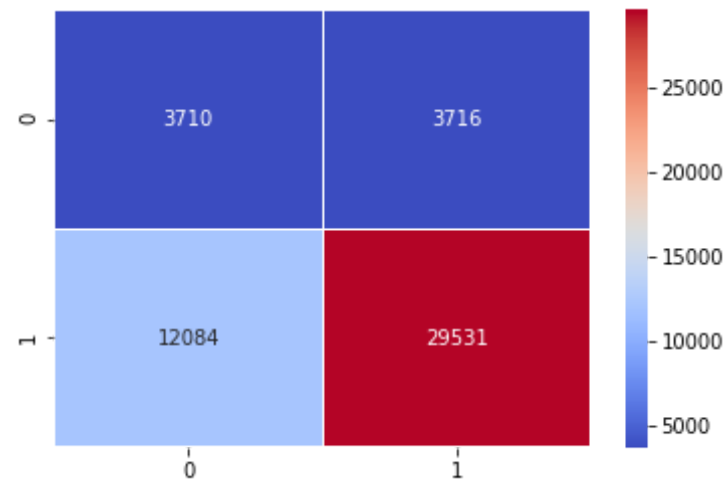


```
, tr_thresholds, train_fpr, train_fpr))

print("Train Confusion Matrix")
sns.heatmap(train_confusion_matrix,annot=True,linewidth = 0.1, cmap='coolwarm', fmt='g')
```

the maximum value of  $tpr*(1-fpr)$  0.249999836795 for threshold 0.833  
Train Confusion Matrix

Out[192]: <matplotlib.axes.\_subplots.AxesSubplot at 0x18a334c22b0>



```
In [193]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html

# Test Confusion Matrix Heatmap

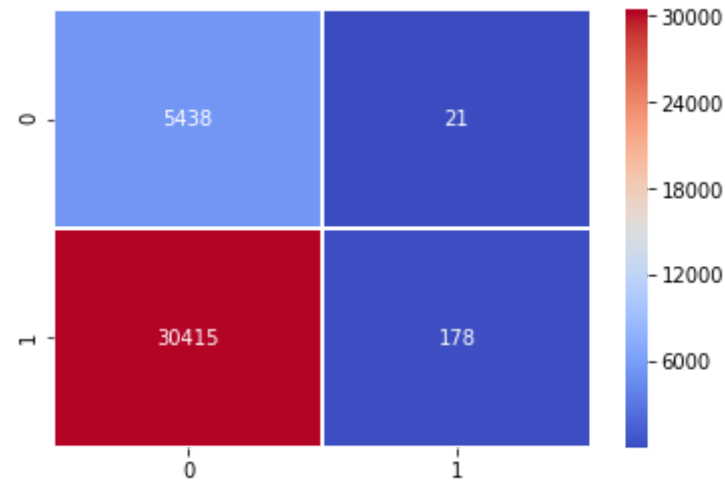
test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, t
r_thresholds, test_fpr, test_fpr))

print("Test Confusion Matrix")
sns.heatmap(test_confusion_matrix,annot=True,linewidth = 0.5, cmap='coolwarm', fmt='g')
```

the maximum value of  $tpr*(1-fpr)$  0.249999991611 for threshold 0.847  
Test Confusion Matrix

Out[193]: <matplotlib.axes.\_subplots.AxesSubplot at 0x18a334c22b0>

Out[193]: <matplotlib.axes.\_subplots.AxesSubplot at 0x18a315af208>



**Set 2 : Categorical, Numerical features + Project\_title(TFIDF) + Preprocessed\_essay (TFIDF with bi-grams with min\_df=10 and max\_features=5000)**

```
In [194]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

# Train data stack
X_tr = hstack((X_Train_categories_one_hot,X_Train_sub_categories_one_hot,
               X_Train_school_state_one_hot,
               X_Train_teacher_prefix_one_hot,X_Train_grade_cat_one_hot,
               X_Train_essay_tfidf,X_Train_titles_tfidf,
               X_train_price_norm)).tocsr()

# CV data Stack
X_cr = hstack((X_CV_categories_one_hot,X_CV_sub_categories_one_hot,X_CV_
               _school_state_one_hot,
               X_CV_teacher_prefix_one_hot,X_CV_grade_cat_one_hot,X_CV_
               essay_tfidf,X_CV_titles_tfidf,
               X_cv_price_norm)).tocsr()
```

```
# Test Data Stack
X_te = hstack((X_Test_categories_one_hot,X_Test_sub_categories_one_hot,
X_Test_school_state_one_hot,
               X_Test_teacher_prefix_one_hot,X_Test_grade_cat_one_hot,X
_Test_essay_tfidf,X_Test_titles_tfidf,
               X_test_price_norm)).tocsr()

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

```
Final Data matrix
(49041, 1907) (49041,)
(24155, 1907) (24155,)
(36052, 1907) (36052,)
```

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

```
In [196]: neigh = LogisticRegression()
parameters = {'C':[0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000,
10000]}
clf = GridSearchCV(neigh, parameters, cv=10, scoring='roc_auc')
clf.fit(X_tr, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_
auc + train_auc_std,alpha=0.2,color='darkblue')

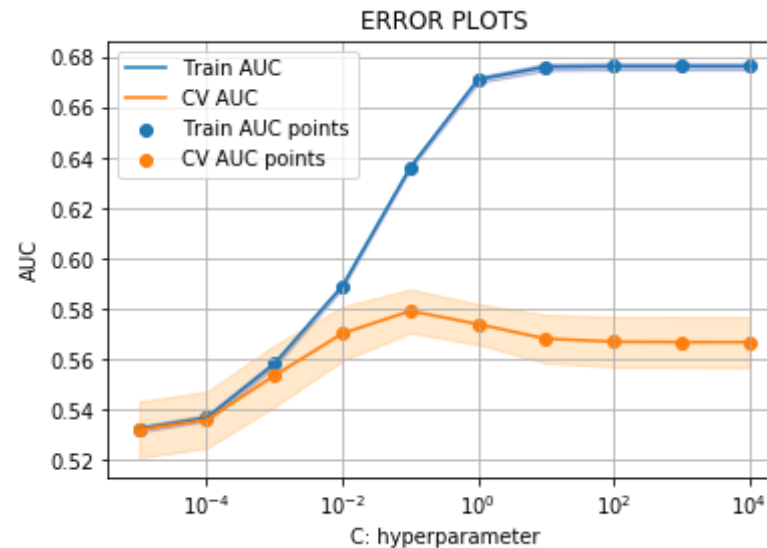
plt.plot(parameters['C'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
```

084039

```
plt.gca().fill_between(parameters['C'], cv_auc - cv_auc_std, cv_auc + cv_auc_std, alpha=0.2, color='darkorange')
```

```
plt.scatter(parameters['C'], train_auc, label='Train AUC points')
plt.scatter(parameters['C'], cv_auc, label='CV AUC points')
```

```
plt.xscale("log")
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



```
In [199]: # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc\_curve.html#sklearn.metrics.roc\_curve
from sklearn.metrics import roc_curve, auc

neigh = LogisticRegression(C=0.5)
neigh.fit(X_tr, y_train)
```

```

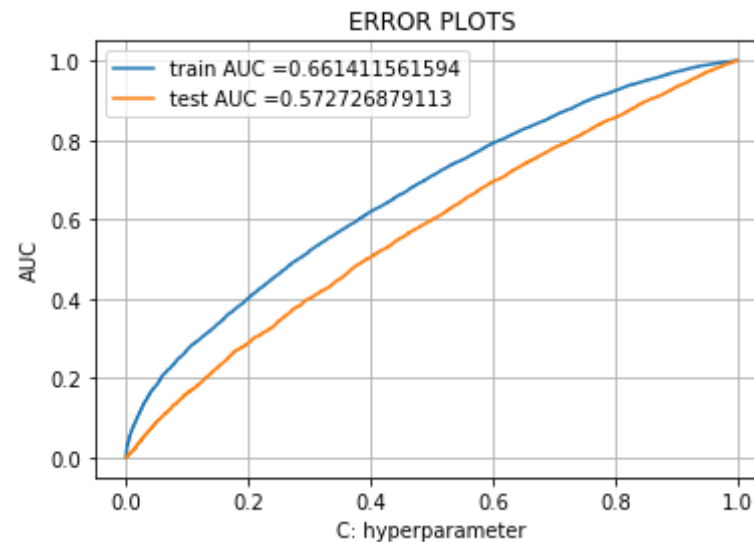
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability
# estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr)
y_test_pred = batch_predict(neigh, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC =" + str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" + str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()

```



```
In [200]: print("="*100)
```

```

from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, tr
ain_fpr, train_fpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test
_fpr, test_fpr)))

```

```

=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.249999981866 for threshold 0.833
[[ 3712  3714]
 [12087 29528]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.249999991611 for threshold 0.847
[[ 4845   614]
 [25221  5372]]

```

```

In [201]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html

# Train Confusion Matrix Heatmap
train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred
, tr_thresholds, train_fpr, train_fpr))

print("Train Confusion Matrix")
sns.heatmap(train_confusion_matrix,annot=True,linewidth = 0.5, cmap='co
olwarm', fmt='g')

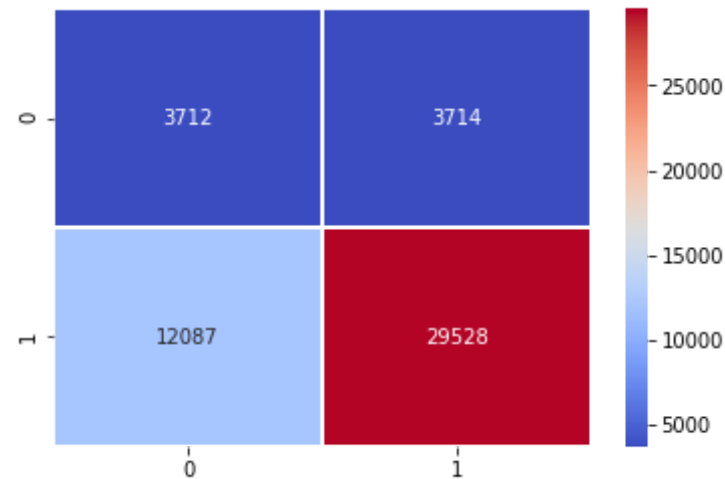
the maximum value of tpr*(1-fpr) 0.249999981866 for threshold 0.833
Train Confusion Matrix

```

```

Out[201]: <matplotlib.axes._subplots.AxesSubplot at 0x18a05961860>

```



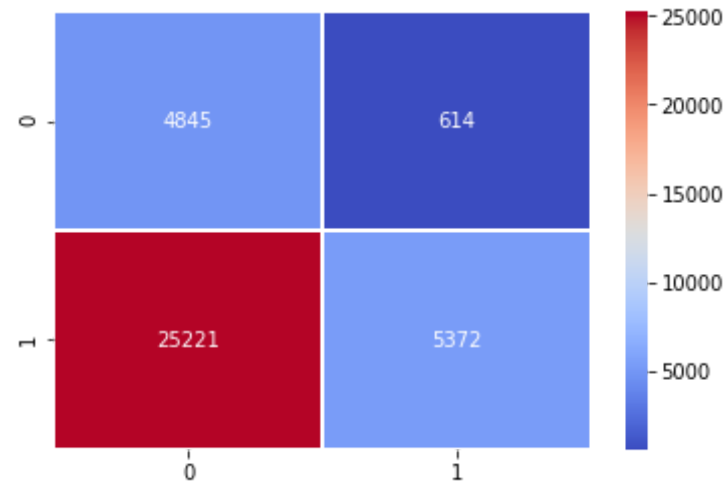
```
In [202]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html
# Test Confusion Matrix Heatmap

test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, t
r_thresholds, test_fpr, test_fpr))

print("Test Confusion Matrix")
sns.heatmap(test_confusion_matrix,annot=True,linewidth = 0.5, cmap='cool
warm', fmt='g')

the maximum value of tpr*(1-fpr) 0.249999991611 for threshold 0.847
Test Confusion Matrix
```

```
Out[202]: <matplotlib.axes._subplots.AxesSubplot at 0x18a610f2cf8>
```



### Set 3 : Categorical, Numerical features + Project\_title(AVG W2V) + Preprocessed\_essay (AVG W2V)

```
In [203]: # Train data stack
X_tr = hstack((X_Train_categories_one_hot,X_Train_sub_categories_one_hot,
X_Train_school_state_one_hot,
               X_Train_teacher_prefix_one_hot,X_Train_grade_cat_one_hot,
X_Train_avg_w2v_vectors,X_Train_avg_w2v_titles_vectors,
               X_train_price_norm)).tocsr()

# CV data Stack
X_cr = hstack((X_CV_categories_one_hot,X_CV_sub_categories_one_hot,X_CV_
_school_state_one_hot,
               X_CV_teacher_prefix_one_hot,X_CV_grade_cat_one_hot,X_CV_
avg_w2v_vectors,X_CV_avg_w2v_titles_vectors,
               X_cv_price_norm)).tocsr()

# Test Data Stack
X_te = hstack((X_Test_categories_one_hot,X_Test_sub_categories_one_hot,
X_Test_school_state_one_hot,
               X_Test_teacher_prefix_one_hot,X_Test_grade_cat_one_hot,X
```



```
_Test_avg_w2v_vectors,X_Test_avg_w2v_titles_vectors,  
X_test_price_norm)).tocsr()
```

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

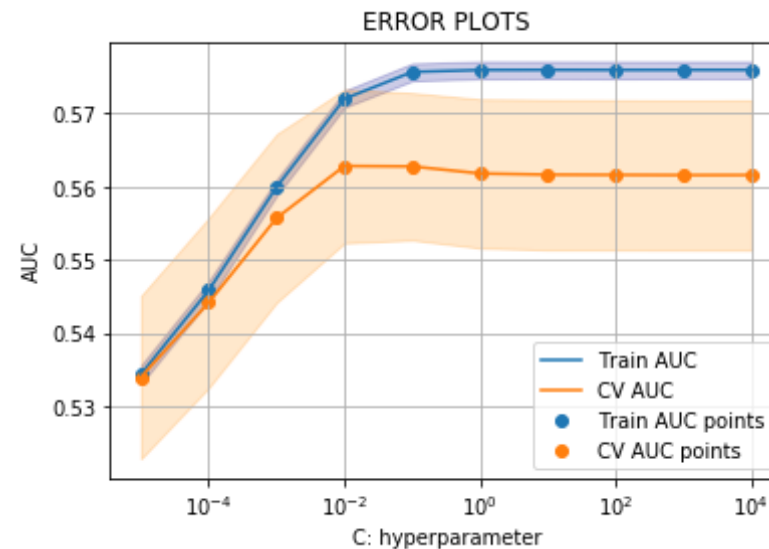
```
Final Data matrix  
(49041, 701) (49041,)  
(24155, 701) (24155,)  
(36052, 701) (36052,)
```

```
=====
```

```
In [204]: neigh = LogisticRegression()  
parameters = {'C':[0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000,  
10000]}  
clf = GridSearchCV(neigh, parameters, cv=10, scoring='roc_auc')  
clf.fit(X_tr, y_train)  
  
train_auc= clf.cv_results_['mean_train_score']  
train_auc_std= clf.cv_results_['std_train_score']  
cv_auc = clf.cv_results_['mean_test_score']  
cv_auc_std= clf.cv_results_['std_test_score']  
  
plt.plot(parameters['C'], train_auc, label='Train AUC')  
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039  
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_  
auc + train_auc_std,alpha=0.2,color='darkblue')  
  
plt.plot(parameters['C'], cv_auc, label='CV AUC')  
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039  
plt.gca().fill_between(parameters['C'],cv_auc - cv_auc_std,cv_auc + cv_  
auc_std,alpha=0.2,color='darkorange')
```

```
plt.scatter(parameters['C'], train_auc, label='Train AUC points')
plt.scatter(parameters['C'], cv_auc, label='CV AUC points')

plt.xscale("log")
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



```
In [216]: # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_
           # _curve.html#sklearn.metrics.roc_curve
           from sklearn.metrics import roc_curve, auc

           neigh = LogisticRegression(C=0.1)
           neigh.fit(X_tr, y_train)
           # roc_auc_score(y_true, y_score) the 2nd parameter should be probabilit
           # y estimates of the positive class
           # not the predicted outputs
```

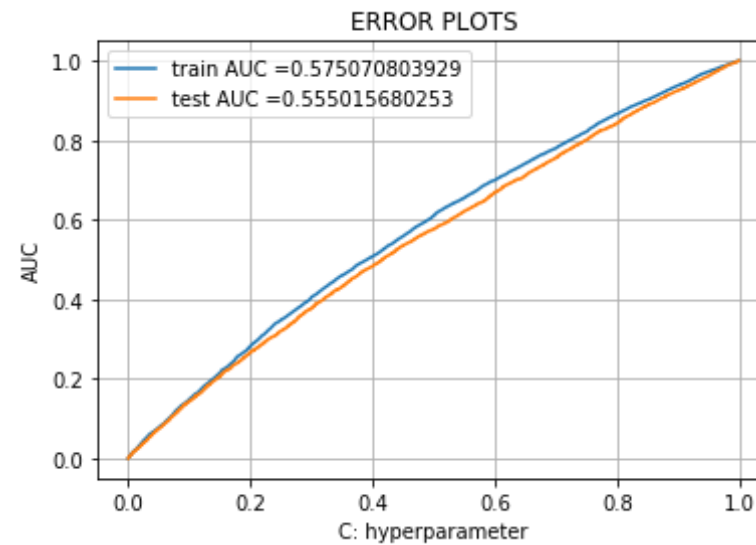
```

y_train_pred = batch_predict(neigh, X_tr)
y_test_pred = batch_predict(neigh, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC =" + str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" + str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()

```



```

In [217]: print("="*100)
           print("Train confusion matrix")
           print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
           print("Test confusion matrix")

```

```
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_fpr)))

=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.844
[[ 3713  3713]
 [16260 25355]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.249999991611 for threshold 0.834
[[ 2043  3416]
 [ 9401 21192]]
```

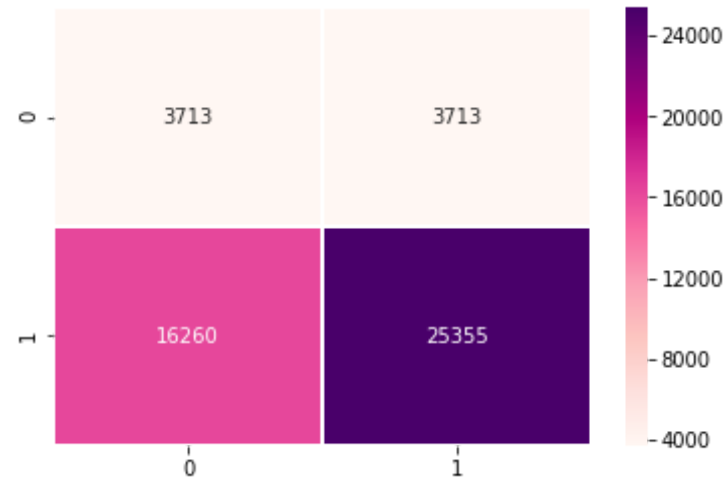
```
In [218]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html
# Train Confusion Matrix Heatmap

train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred,
, tr_thresholds, train_fpr, train_fpr))

print("Train Confusion Matrix")
sns.heatmap(train_confusion_matrix,annot=True,linewidth = 0.5, cmap='RdPu', fmt='g')

the maximum value of tpr*(1-fpr) 0.25 for threshold 0.844
Train Confusion Matrix
```

```
Out[218]: <matplotlib.axes._subplots.AxesSubplot at 0x18a62d921d0>
```



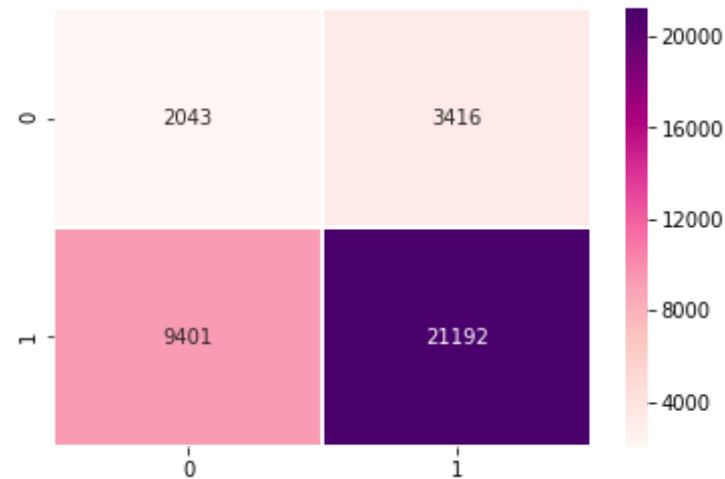
```
In [219]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html
# Test Confusion Matrix Heatmap

test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, t
r_thresholds, test_fpr, test_fpr))

print("Test Confusion Matrix")
sns.heatmap(test_confusion_matrix,annot=True,linewidth = 0.5, cmap='RdPu', fmt='g')

the maximum value of tpr*(1-fpr) 0.249999991611 for threshold 0.834
Test Confusion Matrix
```

```
Out[219]: <matplotlib.axes._subplots.AxesSubplot at 0x18a32ee0278>
```



#### Set 4 : Categorical, Numerical features + Project\_title(TFIDF W2V) + Preprocessed\_essay (TFIDF W2V)

```
In [220]: # Train data stack
X_tr = hstack((X_Train_categories_one_hot,X_Train_sub_categories_one_hot,
X_Train_school_state_one_hot,
               X_Train_teacher_prefix_one_hot,X_Train_grade_cat_one_hot,
X_Train_tfidf_w2v_vectors,X_Train_titles_tfidf_w2v_vectors,
               X_train_price_norm)).tocsr()

# CV data Stack
X_cr = hstack((X_CV_categories_one_hot,X_CV_sub_categories_one_hot,X_CV
_school_state_one_hot,
               X_CV_teacher_prefix_one_hot,X_CV_grade_cat_one_hot,X_CV_
tfidf_w2v_vectors,X_CV_titles_tfidf_w2v_vectors,
               X_cv_price_norm)).tocsr()

# Test Data Stack
X_te = hstack((X_Test_categories_one_hot,X_Test_sub_categories_one_hot,
X_Test_school_state_one_hot,
               X_Test_teacher_prefix_one_hot,X_Test_grade_cat_one_hot,X
```

```

_Test_tfidf_w2v_vectors,X_Test_titles_tfidf_w2v_vectors,
X_test_price_norm)).tocsr()

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

```

```

Final Data matrix
(49041, 701) (49041,)
(24155, 701) (24155,)
(36052, 701) (36052,)
=====
=====

```

```

In [221]: neigh = LogisticRegression()
parameters = {'C':[0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000,
10000]}
clf = GridSearchCV(neigh, parameters, cv=10, scoring='roc_auc')
clf.fit(X_tr, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

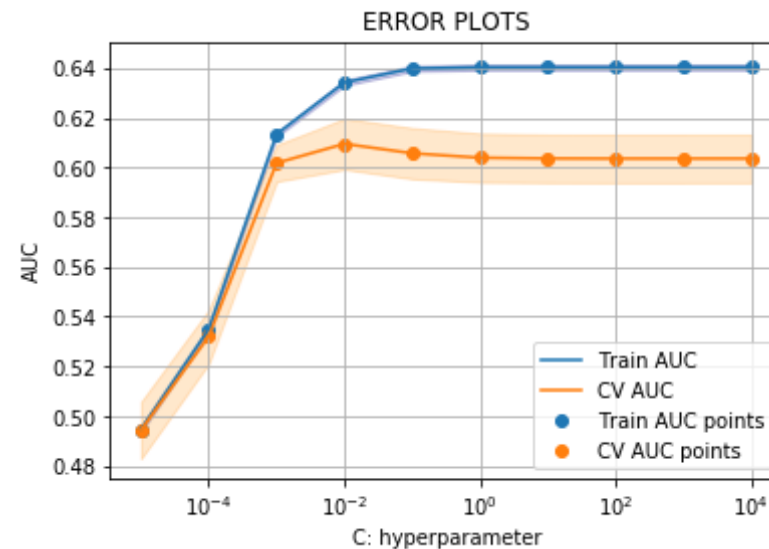
plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_auc + train_auc_std,alpha=0.2,color='darkblue')

plt.plot(parameters['C'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],cv_auc - cv_auc_std,cv_auc + cv_auc_std,alpha=0.2,color='darkorange')

```

```
plt.scatter(parameters['C'], train_auc, label='Train AUC points')
plt.scatter(parameters['C'], cv_auc, label='CV AUC points')

plt.xscale("log")
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



```
In [225]: # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_
           # _curve.html#sklearn.metrics.roc_curve
           from sklearn.metrics import roc_curve, auc

           neigh = LogisticRegression(C=0.05)
           neigh.fit(X_tr, y_train)
           # roc_auc_score(y_true, y_score) the 2nd parameter should be probabilit
           # y estimates of the positive class
           # not the predicted outputs
```



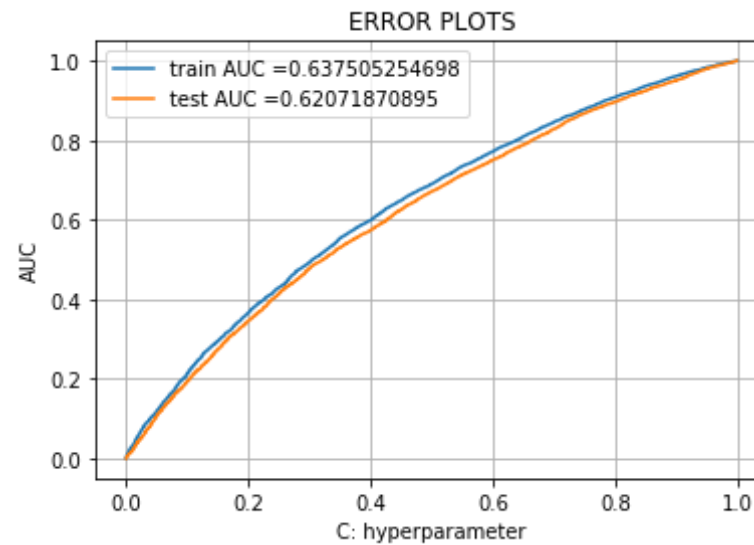
```

y_train_pred = batch_predict(neigh, X_tr)
y_test_pred = batch_predict(neigh, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC =" + str(auc(train_fpr, t
rain_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" + str(auc(test_fpr, test_
tpr)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()

```



```

In [227]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html
          # Train Confusion Matrix Heatmap

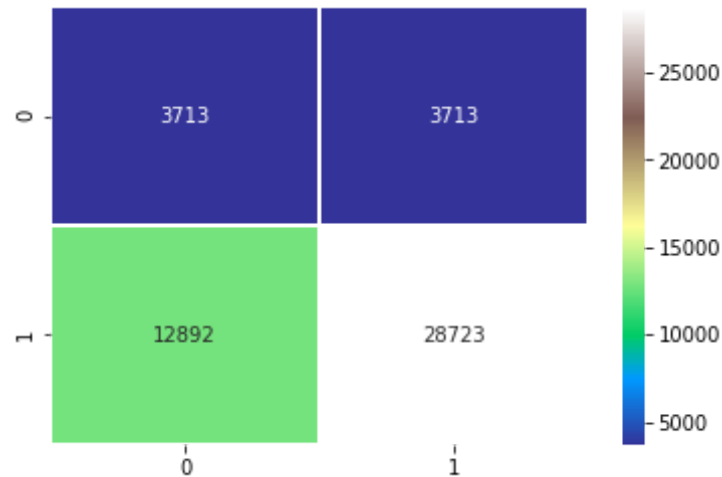
train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred
, tr_thresholds, train_fpr, train_fpr))

```

```
print("Train Confusion Matrix")
sns.heatmap(train_confusion_matrix,annot=True,linewidth = 0.5, cmap='terrain', fmt='g')
```

the maximum value of  $tpr*(1-fpr)$  0.25 for threshold 0.832  
Train Confusion Matrix

Out[227]: <matplotlib.axes.\_subplots.AxesSubplot at 0x18a60d92e80>



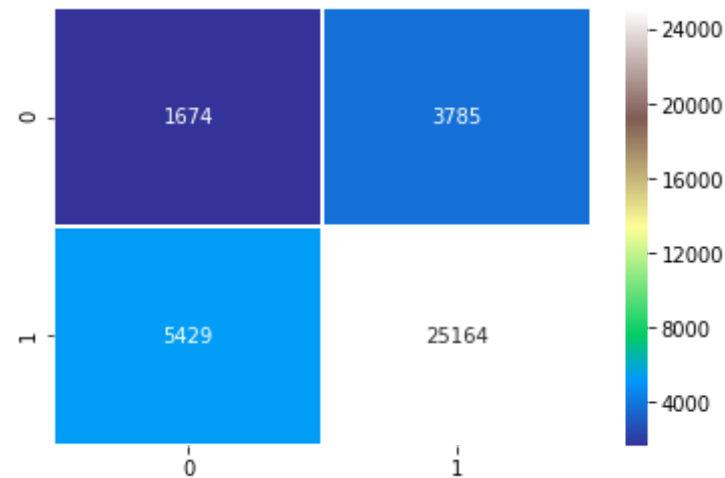
```
In [228]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html
# Test Confusion Matrix Heatmap

test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, t
r_thresholds, test_fpr, test_fpr))

print("Test Confusion Matrix")
sns.heatmap(test_confusion_matrix,annot=True,linewidth = 0.5, cmap='ter
rain',fmt='g')
```

the maximum value of  $tpr*(1-fpr)$  0.249999991611 for threshold 0.854  
Test Confusion Matrix

Out[228]: <matplotlib.axes.\_subplots.AxesSubplot at 0x18a31801400>



## 2.5 Logistic Regression with added Features `Set 5`

```
In [249]: # Train data stack
X_tr = hstack((X_Train_categories_one_hot,X_Train_sub_categories_one_hot,
               X_Train_school_state_one_hot,
               X_Train_teacher_prefix_one_hot,X_Train_grade_cat_one_hot,
               title_word_count_train,
               essay_word_count_train, essay_sent_pos_train, essay_sent_neg_train,
               essay_sent_neu_train,
               essay_sent_comp_train)).tocsr()

# CV data Stack
X_cr = hstack((X_CV_categories_one_hot,X_CV_sub_categories_one_hot,X_CV_school_state_one_hot,
               X_CV_teacher_prefix_one_hot,X_CV_grade_cat_one_hot,title_word_count_cv,
               essay_word_count_cv, essay_sent_pos_cv, essay_sent_neg_cv,
               essay_sent_neu_cv,
               essay_sent_comp_cv)).tocsr()

# Test Data Stack
```

```

X_te = hstack((X_Test_categories_one_hot,X_Test_sub_categories_one_hot,
X_Test_school_state_one_hot,
               X_Test_teacher_prefix_one_hot,X_Test_grade_cat_one_hot,t
itle_word_count_test,
               essay_word_count_test, essay_sent_pos_test, essay_sent_n
eg_test, essay_sent_neu_test,
               essay_sent_comp_test)).tocsr()

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

```

```

Final Data matrix
(49041, 106) (49041,)
(24155, 106) (24155,)
(36052, 106) (36052,)

```

```

=====
=====

```

```

In [250]: neigh = LogisticRegression()
parameters = {'C':[0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000,
10000]}
clf = GridSearchCV(neigh, parameters, cv=10, scoring='roc_auc')
clf.fit(X_tr, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']

plt.plot(parameters['C'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'],train_auc - train_auc_std,train_
auc + train_auc_std,alpha=0.2,color='darkblue')

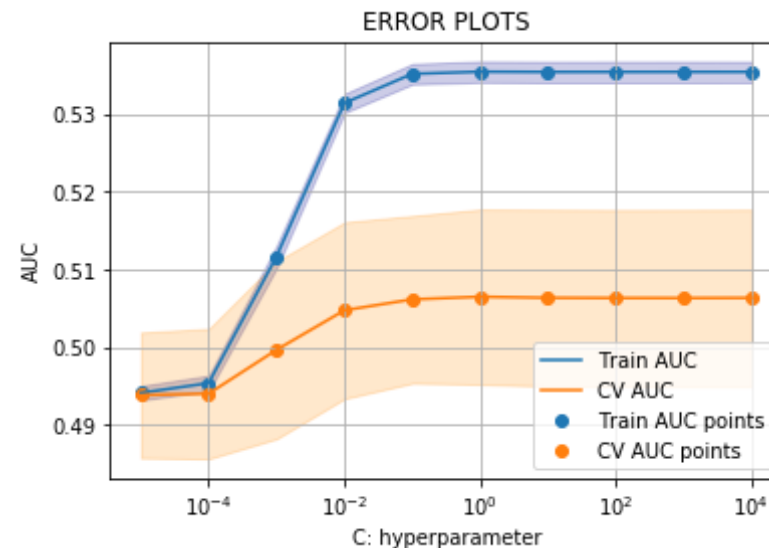
plt.plot(parameters['C'], cv_auc, label='CV AUC')

```

```
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['C'], cv_auc - cv_auc_std, cv_auc + cv_auc_std, alpha=0.2, color='darkorange')

plt.scatter(parameters['C'], train_auc, label='Train AUC points')
plt.scatter(parameters['C'], cv_auc, label='CV AUC points')

plt.xscale("log")
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



```
In [259]: # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc\_curve.html#sklearn.metrics.roc\_curve
from sklearn.metrics import roc_curve, auc

neigh = LogisticRegression(C=0.001)
```

```

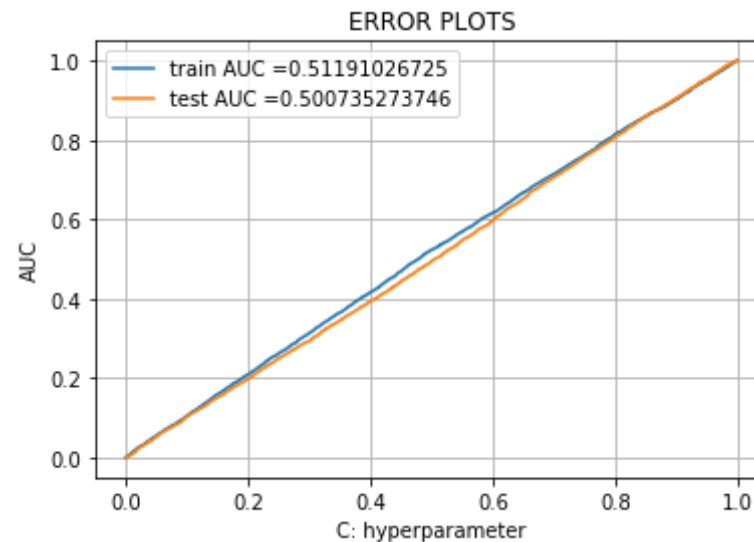
neigh.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability
# estimates of the positive class
# not the predicted outputs

y_train_pred = batch_predict(neigh, X_tr)
y_test_pred = batch_predict(neigh, X_te)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)

plt.plot(train_fpr, train_tpr, label="train AUC =" + str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC =" + str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()

```



```
In [260]: print("="*100)
          from sklearn.metrics import confusion_matrix
          print("Train confusion matrix")
          print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, tr
            ain_fpr, train_fpr)))
          print("Test confusion matrix")
          print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test
            _fpr, test_fpr)))
```

```
=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.845
[[ 3713  3713]
 [19837 21778]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.249999991611 for threshold 0.846
[[ 3225  2234]
 [18320 12273]]
```

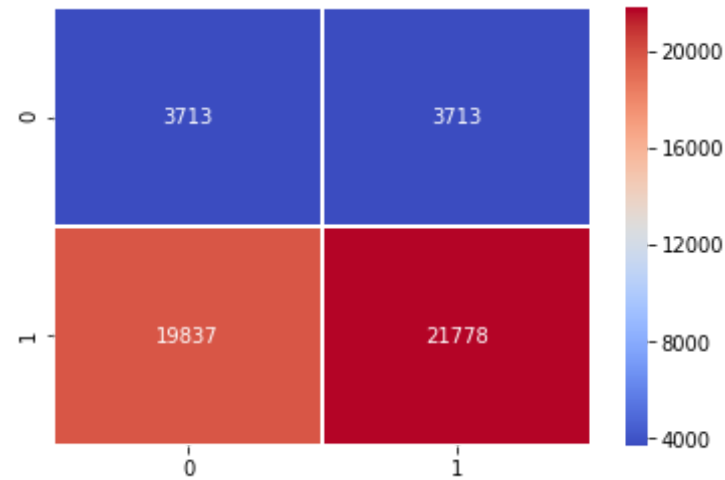
```
In [261]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html

          # Train Confusion Matrix Heatmap
          train_confusion_matrix = confusion_matrix(y_train, predict(y_train_pred
            , tr_thresholds, train_fpr, train_fpr))

          print("Train Confusion Matrix")
          sns.heatmap(train_confusion_matrix,annot=True,linewidth = 0.5, cmap='co
            olwarm', fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.845
Train Confusion Matrix
```

```
Out[261]: <matplotlib.axes._subplots.AxesSubplot at 0x18a53278160>
```



```
In [262]: # https://seaborn.pydata.org/generated/seaborn.heatmap.html
# Test Confusion Matrix Heatmap

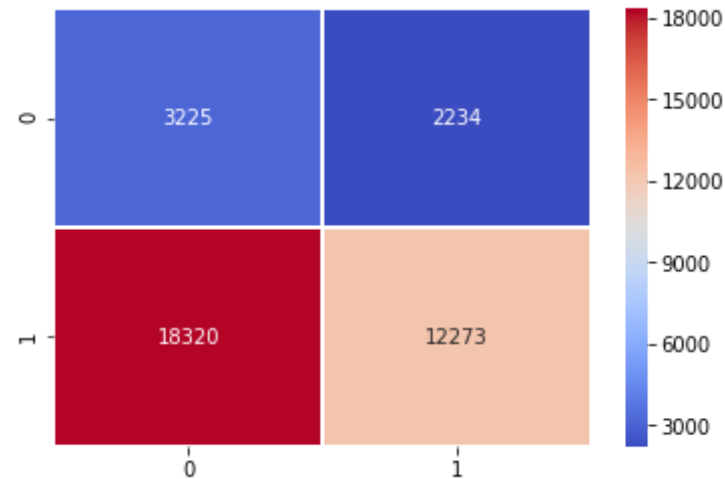
test_confusion_matrix = confusion_matrix(y_test, predict(y_test_pred, t
r_thresholds, test_fpr, test_fpr))

print("Test Confusion Matrix")
sns.heatmap(test_confusion_matrix,annot=True,linewidth = 0.5, cmap='cool
warm', fmt='g')

the maximum value of tpr*(1-fpr) 0.249999991611 for threshold 0.846
Test Confusion Matrix
```

```
Out[262]: <matplotlib.axes._subplots.AxesSubplot at 0x18a612eec50>
```





### 3. Conclusion

```
In [265]: # http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Featurization", "Best C Value", "Train AUC", "Test AU
C"]

x.add_row(["Logistic Regression on BOW", 0.5, 0.66, 0.57])
x.add_row(["Logistic Regression on TFIDF", 0.5, 0.66, 0.57])
x.add_row(["Logistic Regression on AVG W2V", 0.1, 0.57, 0.55])
x.add_row(["Logistic Regression on TFIDF W2V", 0.05, 0.64, 0.62])
x.add_row(["Logistic Regression on Categorical features, Numerical feat
ures & Essay Sentiments", 0.001, 0.51, 0.50])

print(x)
```

+-----

Featurization				
	Best C Value	Train AUC	Test AUC	
Logistic Regression on BOW				
	0.5	0.66	0.57	
Logistic Regression on TFIDF				
	0.5	0.66	0.57	
Logistic Regression on AVG W2V				
	0.1	0.57	0.55	
Logistic Regression on TFIDF W2V				
	0.05	0.64	0.62	
Logistic Regression on Categorical features, Numerical features & Essay Sentiments	0.001	0.51	0.5	