

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	Descri
<code>project_id</code>	A unique identifier for the proposed project. Example: p03
<code>project_title</code>	Title of the project. Exam Art Will Make You Ha First Grade
<code>project_grade_category</code>	Grade level of students for which the project is targeted. One of the foll enumerated va Grades Pr Grades Grades Grades
<code>project_subject_categories</code>	One or more (comma-separated) subject categories for the project fro following enumerated list of va Applied Lear Care & Hu Health & Sp History & Ci Literacy & Lang Math & Sci Music & The Special N Wa
<code>school_state</code>	State where school is located (Two-letter U.S. postal (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_co) Example Music & The Literacy & Language, Math & Sci
<code>project_subject_subcategories</code>	One or more (comma-separated) subject subcategories for the pr Exam Lite Literature & Writing, Social Scie
<code>project_resource_summary</code>	An explanation of the resources needed for the project. Exam My students need hands on literacy materials to mar sensory ne
<code>project_essay_1</code>	First application e
<code>project_essay_2</code>	Second application e
<code>project_essay_3</code>	Third application e
<code>project_essay_4</code>	Fourth application e

Feature	Description
project_submitted_datetime	Datetime when project application was submitted. Example: 2016-04-12:43:56
teacher_id	A unique identifier for the teacher of the proposed project. Example: bdf8baa8fedef6bfeec7ae4ff1c1
teacher_prefix	Teacher's title. One of the following enumerated values: <ul style="list-style-type: none"> • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same teacher. Example: 1

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

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

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

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

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

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



Notes on the Essay Data

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

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

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

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

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

```
In [0]: %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
```

```
D:\installed\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

```
In [0]: project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

```
In [0]: 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 [0]: 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[0]:

	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 [0]: 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 [0]: 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 [0]: # 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 [0]: project_data.head(2)
```

Out[0]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_:
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bcd1151f324dd63a	Mr.	FL	

```
In [0]: ##### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

```
In [0]: # 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)
print(project_data['essay'].values[99999])
print("="*50)
```

My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 languages represented in our English Learner program with students at every level of mastery. We also have over 40 countries represented with the families within our school. Each student brings a wealth of knowledge and experiences to us that open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits of your world.\"-Ludwig Wittgenstein Our English 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 along side of their children. Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English language even if no one at home is able to assist. All families with students within the Level 1 proficiency status, will be offered to be a part of this program. These educational videos will be specially chosen by the English Learner Teacher and will be sent home regularly to watch. The videos are to help the child develop early reading skills.\r\n\r\nParents that do not have access to a dvd player will have the opportunity to check out a dvd player to use for the year. The plan is to use these videos and educational dvd's for the years to come for other EL students.\r\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\n\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 students to do desk work and move at the same time. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their core muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit still.nnnannan

=====

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 theme

d 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\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan

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

The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates. The great teacher inspires. -William A. Ward\r\n\r\n\r\nMy school has 803 students which is makeup is 97.6% African-American, making up the largest segment of the student body. A typical school in Dallas is made up of 23.2% African-American students. Most of the students are on free or reduced lunch. We aren't receiving doctors, lawyers, or engineers children from rich backgrounds or neighborhoods. As an educator I am inspiring minds of young children and we focus not only on academics but one smart, effective, efficient, and disciplined students with good character. In our classroom we can utilize the Bluetooth for swift transitions during class. I use a speaker which doesn't amplify the sound enough to receive the message. Due to the volume of my speaker my students can't hear videos or books clearly and it isn't making the lessons as meaningful. But with the bluetooth speaker my students will be able to hear and I can stop, pause and replay it at any time.\r\n\r\nThe cart will allow me to have more room for storage of things that are needed for the day and has an extra part to it I can use. The table top chart has all of the letter, words and pictures for students to learn about different letters and it

is more accessible.nannan

=====

```
In [0]: # 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 [0]: 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. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore. Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say. Wobble chairs are the answer and I love them because they develop their core, which enhances gross motor and 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 [0]: # \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. 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 [0]: #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 [0]: # gist://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', 'it
self', 'they', 'them', 'their', \
, 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "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', 'becau
se', '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', 'a
ll', 'any', 'both', 'each', 'few', 'more', \
, 'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
, 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d'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", 'm
a', 'mightn', "mightn't", 'mustn', \
, 'mustn't', 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
, 'won', "won't", 'wouldn', "wouldn't"]
```

```
In [0]: # Combining all the above students
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 [00:59<00:00, 1842.74it/s]
```

```
In [0]: # after preprocessing  
preprocessed_essays[20000]
```

```
Out[0]: 'my kindergarten students varied disabilities ranging speech language delays  
cognitive delays gross fine motor delays autism they eager beavers always str  
ive work hardest working past limitations the materials ones i seek students  
i teach title i school students receive free reduced price lunch despite disa  
bilities limitations students love coming school come eager learn explore hav  
e ever felt like ants pants needed groove move meeting this kids feel time th  
e want able move learn say wobble chairs answer i love develop core enhances  
gross motor turn fine motor skills they also want learn games kids not want s  
it worksheets they want learn count jumping playing physical engagement key s  
uccess the number toss color shape mats make happen my students forget work f  
un 6 year old deserves nannan'
```

1.4 Preprocessing of `project_title`

```
In [0]: # similarly you can preprocess the titles also
```

1.5 Preparing data for models

```
In [0]: project_data.columns
```

```
Out[0]: Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',  
              'project_submitted_datetime', 'project_grade_category', 'project_titl  
e',  
              '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/> (<https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorical-and-numerical-features/>)

```
In [0]: # 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 [0]: # 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 encoding ", sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encoding (109248, 30)
```

```
In [0]: # you can do the similar thing with state, teacher_prefix and project_grade_category also
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

```
In [0]: # We are considering only the words which appeared in at least 10 documents(ros 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 [0]: # you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

1.5.2.2 TFIDF vectorizer

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

# =====
Output:

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

# =====

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

for i in preproced_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 our coup
us", \
      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[0]: '''# Reading glove vectors in python: https://stackoverflow.com/a/38230349/40
84039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n
f = open(gloveFile,\r', encoding="utf8")\n    model = {}\n    for line in t
qdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n
embedding = np.array([float(val) for val in splitLine[1:]])\n        model[word]
= embedding\n    print ("Done.",len(model)," words loaded!")\n    return
model\nmodel = loadGloveModel('glove.42B.300d.txt')\n\n# =====
=====
\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/
s]\nDone. 1917495 words loaded!\n\n# =====
\n\nwords =
[]\nfor i in preprocod_texts:\n    words.extend(i.split(' '))\n\nfor i in p
reprocod_titles:\n    words.extend(i.split(' '))\n\nprint("all the words in t
he coupus", len(words))\n\nwords = set(words)\n\nprint("the unique words in the c
oupus", len(words))\n\ninter_words = set(model.keys()).intersection(words)\n\np
rint("The number of words that are present in both glove vectors and our coup
us", len(inter_words), "(", np.round(len(inter_words)/len(words)*100,
3), "%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in wor
ds:\n    if i in words_glove:\n        words_courpus[i] = model[i]\n\nprint("wo
rd 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-v
ariables-in-python/\n\nimport pickle\nwith open('glove_vectors', 'wb') as
f:\n    pickle.dump(words_courpus, f)\n\n\n'

```

```

In [0]: # 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 [0]: # 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 [00:32<00:00, 3369.33it/s]

```

```

109248
300

```



```
In [0]: # check this one: https://www.youtube.com/watch?v=0H0qOcln3Z4&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=[725.05 213.03 32
9. ... 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(price_scalar.var_[0])}")

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

```
In [0]: price_standardized
```

```
Out[0]: array([[0.00098843, 0.00191166, 0.00330448, ..., 0.00153418, 0.00046704,
0.00070265]])
```

1.5.4 Merging all the above features

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

```
In [0]: 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 [0]: # 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[0]: (109248, 16663)
```

```
In [0]: # please write all the code with proper documentation, and proper titles for each subsection
# when you plot any graph make sure you use
# a. Title, that describes your plot, this will be very helpful to the reader
# b. Legends if needed
# c. X-axis label
# d. Y-axis label
```

Computing Sentiment Scores

```

In [0]: 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 teach 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 americans o
        ur school is a caring community of successful \
        learners which can be seen through collaborative student project based learnin
        g in and out of the classroom kindergarteners \
        in my class love to work with hands on materials and have many different oppor
        tunities to practice a skill before it is\
        mastered having the social skills to work cooperatively with friends is a cruc
        ial aspect of the kindergarten curriculum\
        montana is the perfect place to learn about agriculture and nutrition my stude
        nts love to role play in our pretend kitchen\
        in the early childhood classroom i have had several kids ask me can we try coo
        king with real food i will take their idea \
        and create common core cooking lessons where we learn important math and writi
        ng concepts while cooking delicious healthy \
        food for snack time my students will have a grounded appreciation for the work
        that went into making the food and knowledge \
        of where the ingredients came from as well as how it is healthy for their bodi
        es this project would expand our learning of \
        nutrition and agricultural cooking recipes by having us peel our own apples to
        make homemade applesauce make our own bread \
        and mix up healthy plants from our classroom garden in the spring we will also
        create our own cookbooks to be printed and \
        shared with families students will gain math and literature skills as well 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

```

D:\installed\Anaconda3\lib\site-packages\nltk\twitter__init__.py:20: UserWarning:

The twython library has not been installed. Some functionality from the twitter package will not be available.

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

Assignment 7: SVM

1. [Task-1] Apply Support Vector Machines(SGDClassifier with hinge loss: Linear SVM) on these feature sets

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

2. The hyper parameter tuning (best alpha in range $[10^{-4}$ to 10^4], and the best penalty among 'l1', 'l2')

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

3. Representation of results

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



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



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



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

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

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

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

4. [Task-2] Apply the Support Vector Machines on these features by finding the best hyper parameter as suggested in step 2 and step 3 (<https://seaborn.pydata.org/generated/seaborn.heatmap.html>)

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

- Consider these set of features **Set 5** : (<https://seaborn.pydata.org/generated/seaborn.heatmap.html>)
 - **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
(<https://seaborn.pydata.org/generated/seaborn.heatmap.html>)
- **Apply** (<https://seaborn.pydata.org/generated/seaborn.heatmap.html>) **TruncatedSVD** (<http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html>) on **TfidfVectorizer** (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) of essay text, choose the number of components (`n_components`) using **elbow method** (<https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/pca-code-example-using-non-visualization/>) : numerical data

- **Conclusion**

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



Note: Data Leakage

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

2. Support Vector Machines

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

```

In [1]: %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_notebook as tqdm1
from tqdm import tqdm
import time
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

from sklearn.model_selection import train_test_split

```

```

C:\Users\LENOVO\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress')
C:\Users\LENOVO\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")

```

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

```
In [3]: print("Number of data points in train data", project_data.shape)
print('-'*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']

Text preprocessing(1)

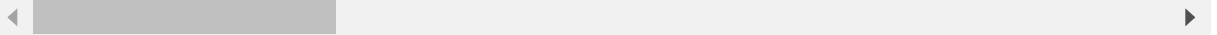
```
In [4]: categories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039

# 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())
```

```
In [5]: project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(5)
```

Out[5]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_:
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	



```
In [6]: # count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
my_counter
```

```
Out[6]: Counter({'Literacy_Language': 52239,
                 'History_Civics': 5914,
                 'Health_Sports': 14223,
                 'Math_Science': 41421,
                 'SpecialNeeds': 13642,
                 'AppliedLearning': 12135,
                 'Music_Arts': 10293,
                 'Warmth': 1388,
                 'Care_Hunger': 1388})
```

```
In [7]: # dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))

# ind = np.arange(len(sorted_cat_dict))
# plt.figure(figsize=(20,5))
# p1 = plt.bar(ind, list(sorted_cat_dict.values()))

# plt.ylabel('Projects')
# plt.title('% of projects aproved category wise')
# plt.xticks(ind, list(sorted_cat_dict.keys()))
# plt.show()
# print(sorted_cat_dict)
```

```
In [8]: 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-fr
# om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
# g-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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory 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 placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
            temp +=j.strip()+" #" "abc ".strip() will return "abc", remove the tra
iling spaces
            temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

```
In [9]: project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[9]:

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

```
In [10]: # count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

```
In [11]: # dict sort by value python: https://stackoverflow.com/a/613218/4084039
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))

# ind = np.arange(len(sorted_sub_cat_dict))
# plt.figure(figsize=(20,5))
# p1 = plt.bar(ind, list(sorted_sub_cat_dict.values()))

# plt.ylabel('Projects')
# plt.title('% of projects aproved state wise')
# plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
# plt.show()
```

```
In [12]: # 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 [13]: # https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indices-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'})
price_data.reset_index()
price_data.head(2)
```

Out[13]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

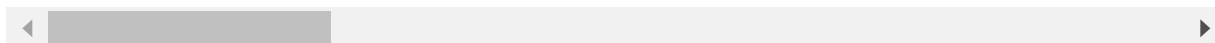
```
In [14]: # join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

```
In [15]: #presence of the numerical digits in a strings with numeric : https://stackoverflow.com/a/19859308/8089731
def hasNumbers(inputString):
    return any(i.isdigit() for i in inputString)
p1 = project_data[['id', 'project_resource_summary']]
p1 = pd.DataFrame(data=p1)
p1.columns = ['id', 'digits_in_summary']
p1['digits_in_summary'] = p1['digits_in_summary'].map(hasNumbers)
# https://stackoverflow.com/a/17383325/8089731
p1['digits_in_summary'] = p1['digits_in_summary'].astype(int)
project_data = pd.merge(project_data, p1, on='id', how='left')
project_data.head(5)
```

Out[15]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

5 rows × 21 columns



Text preprocessing(2)

In [16]: [# https://stackoverflow.com/a/47091490/4084039](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 [17]: [# https://gist.github.com/sebleier/554280](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', 'it
self', 'they', 'them', 'their', \
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "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', 'becau
se', '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', 'a
ll', 'any', 'both', 'each', 'few', 'more', \
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d'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", 'm
a', 'mightn', "mightn't", 'mustn', \
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

```
In [18]: # Combining all the above statemennts
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('\n', ' ')
    sent = sent.replace('\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    sent = re.sub('nannan', '', 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 [00:59<00:00, 1850.04it/s]
```

```
In [19]: from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for title in tqdm(project_data['project_title'].values):
    _title = decontracted(title)
    _title = _title.replace('\r', ' ')
    _title = _title.replace('\n', ' ')
    _title = _title.replace('\n', ' ')
    _title = re.sub('[^A-Za-z0-9]+', ' ', _title)
    # https://gist.github.com/sebleier/554280
    _title = ' '.join(e for e in _title.split() if e not in stopwords)
    preprocessed_titles.append(_title.lower().strip())
```

```
100%|████████████████████████████████████████████████████████████████████████████████|
109248/109248 [00:02<00:00, 37735.42it/s]
```

```
In [20]: preprocessed_titles[1000]
```

```
Out[20]: 'sailing into super 4th grade year'
```

```

In [21]: project_grade_catogories = list(project_data['project_grade_category'].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

project_grade_cat_list = []
for i in tqdm1(project_grade_catogories):
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory 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 placeing 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('&','_')
    project_grade_cat_list.append(temp.strip())

```

```

In [22]: project_data['clean_project_grade_category'] = project_grade_cat_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data.head(2)

```

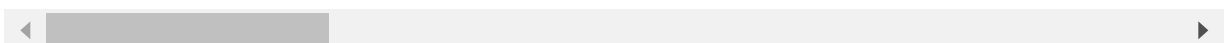
Out[22]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
--	------------	----	------------	----------------	--------------	----------

0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
---	--------	---------	----------------------------------	------	----	--

1	140945	p258326	897464ce9ddc600bcd1151f324dd63a	Mr.	FL	
---	--------	---------	---------------------------------	-----	----	--

2 rows × 21 columns



```
In [23]: project_data.drop(['project_essay_1', 'project_essay_2', 'project_essay_3', 'project_essay_4'], axis=1, inplace=True)
project_data.head(2)
```

Out[23]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bcd1151f324dd63a	Mr.	FL	



```
In [24]: #Replacing Nan's with maximum occured value: https://stackoverflow.com/a/51053916/8089731
project_data['teacher_prefix'].value_counts().argmax()
project_data.fillna(value=project_data['teacher_prefix'].value_counts().argmax(), axis=1, inplace=True)
```

```
In [25]: project_data['preprocessed_essays'] = preprocessed_essays
project_data['preprocessed_titles'] = preprocessed_titles
```

```
In [26]: project_data.columns
```

```
Out[26]: Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
               'project_submitted_datetime', 'project_title',
               'project_resource_summary',
               'teacher_number_of_previously_posted_projects', 'project_is_approved',
               'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
               'digits_in_summary', 'clean_project_grade_category',
               'preprocessed_essays', 'preprocessed_titles'],
              dtype='object')
```

2.2 Make Data Model Ready: encoding numerical, categorical features

```
In [27]: X_train, X_test, y_train, y_test = train_test_split(project_data,project_data[
'project_is_approved'], test_size=0.33, stratify = project_data['project_is_ap
proved'])
# X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=
0.33, stratify=y_train)

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

```
(73196, 18)
```

```
(36052, 18)
```

1.4.1 Vectorizing Categorical data

```
In [28]: # we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowe
rcase=False, binary=True)
vectorizer_cat.fit(X_train['clean_categories'].values)
print(vectorizer_cat.get_feature_names())

categories_one_hot_train = vectorizer_cat.transform(X_train['clean_categories'
].values)
# categories_one_hot_cv = vectorizer_cat.transform(X_cv['clean_categories'].va
lues)
categories_one_hot_test = vectorizer_cat.transform(X_test['clean_categories'].
values)
print("Shape of matrix after one hot encodig_train ",categories_one_hot_train.
shape)
# print("Shape of matrix after one hot encodig_cv ",categories_one_hot_cv.shap
e)
print("Shape of matrix after one hot encodig_test ",categories_one_hot_test.sh
ape)

['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig_train (73196, 9)
Shape of matrix after one hot encodig_test (36052, 9)
```

```

In [29]: # we use count vectorizer to convert the values into one hot encoded features
vectorizer_sub_cat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_sub_cat.fit(X_train['clean_subcategories'].values)
print(vectorizer_sub_cat.get_feature_names())

sub_categories_one_hot_train = vectorizer_sub_cat.transform(X_train['clean_subcategories'].values)
# sub_categories_one_hot_cv = vectorizer_sub_cat.transform(X_cv['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer_sub_cat.transform(X_test['clean_subcategories'].values)
print("Shape of matrix after one hot encoding_train ", sub_categories_one_hot_train.shape)
# print("Shape of matrix after one hot encoding_cv ", sub_categories_one_hot_cv.shape)
print("Shape of matrix after one hot encoding_test ", sub_categories_one_hot_test.shape)

['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encoding_train (73196, 30)
Shape of matrix after one hot encoding_test (36052, 30)

```



```
In [30]: # we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_state = CountVectorizer( lowercase=False, binary=True)
vectorizer_state.fit(X_train['school_state'].values)
print(vectorizer_state.get_feature_names())

school_state_one_hot_train = vectorizer_state.transform(X_train['school_state'].values)
# school_state_one_hot_cv = vectorizer_state.transform(X_cv['school_state'].values)
school_state_one_hot_test = vectorizer_state.transform(X_test['school_state'].values)
print("Shape of matrix after one hot encoding_train ", school_state_one_hot_train.shape)
# print("Shape of matrix after one hot encoding_cv ", school_state_one_hot_cv.shape)
print("Shape of matrix after one hot encoding_test ", school_state_one_hot_test.shape)

['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 encoding_train (73196, 51)
Shape of matrix after one hot encoding_test (36052, 51)
```

```
In [31]: # we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_teacherprefix = CountVectorizer( lowercase=False, binary=True)
vectorizer_teacherprefix.fit(X_train['teacher_prefix'].values.astype('U'))
print(vectorizer_teacherprefix.get_feature_names())

#https://stackoverflow.com/a/39308809/8089731
teacher_prefix_one_hot_train = vectorizer_teacherprefix.transform(X_train['teacher_prefix'].values.astype('U'))
# teacher_prefix_one_hot_cv = vectorizer_teacherprefix.transform(X_cv['teacher_prefix'].values.astype('U'))
teacher_prefix_one_hot_test = vectorizer_teacherprefix.transform(X_test['teacher_prefix'].values.astype('U'))
print("Shape of matrix after one hot encoding_train ", teacher_prefix_one_hot_train.shape)
# print("Shape of matrix after one hot encoding_cv ", teacher_prefix_one_hot_cv.shape)
print("Shape of matrix after one hot encoding_test ", teacher_prefix_one_hot_test[:5,:])
# print(X_train['teacher_prefix'].value_counts())

['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encoding_train (73196, 5)
Shape of matrix after one hot encoding_test (0, 1) 1
(1, 2) 1
(2, 3) 1
(3, 3) 1
(4, 2) 1
```

```
In [32]: print(project_data['clean_project_grade_category'].unique())
```

```
['GradesPreK-2' 'Grades6-8' 'Grades3-5' 'Grades9-12']
```

```
In [33]: # we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
# https://stackoverflow.com/a/38161028/8089731
pattern = "(?u)\\b[\\w-]+\\b"
vectorizer_projectgrade = CountVectorizer(token_pattern=pattern, lowercase=False, binary=True)
vectorizer_projectgrade.fit(X_train['clean_project_grade_category'].values)
print(vectorizer_projectgrade.get_feature_names())
```

```
#https://stackoverflow.com/a/39308809/8089731
project_grade_category_one_hot_train = vectorizer_projectgrade.transform(X_train['clean_project_grade_category'].values)
# project_grade_category_one_hot_cv = vectorizer_projectgrade.transform(X_cv['clean_project_grade_category'].values)
project_grade_category_one_hot_test = vectorizer_projectgrade.transform(X_test['clean_project_grade_category'].values)
print("Shape of matrix after one hot encoding_train ", project_grade_category_one_hot_train.shape)
# print("Shape of matrix after one hot encoding_cv ", project_grade_category_one_hot_cv.shape)
print("Shape of matrix after one hot encoding_test ", project_grade_category_one_hot_test[:5,:])
```

```
['Grades3-5', 'Grades6-8', 'Grades9-12', 'GradesPreK-2']
Shape of matrix after one hot encoding_train (73196, 4)
Shape of matrix after one hot encoding_test (0, 0) 1
(1, 0) 1
(2, 2) 1
(3, 1) 1
(4, 3) 1
```

Vectorizing Numerical features

```

In [34]: # check this one: https://www.youtube.com/watch?v=0H0q0cLn3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/s
klearn.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=[725.05 213.03 32
9. ... 399. 287.73 5.5 ].
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(X_train['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(price_sc
alar.var_[0])}")

# Now standardize the data with above maen and variance.
price_standardized_train = price_scalar.transform(X_train['price'].values.resh
ape(-1, 1))
# price_standardized_cv = price_scalar.transform(X_cv['price'].values.reshape
(-1, 1))
price_standardized_test = price_scalar.transform(X_test['price'].values.reshap
e(-1, 1))
print(price_standardized_train.shape)
# print(price_standardized_cv.shape)
print(price_standardized_test.shape)

```

```

Mean : 299.12005478441444, Standard deviation : 372.8621877359348
(73196, 1)
(36052, 1)

```

```
In [35]: # check this one: https://www.youtube.com/watch?v=0H0q0cLn3Z4&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=[725.05 213.03 32
9. ... 399. 287.73 5.5 ].
# Reshape your data either using array.reshape(-1, 1)

quantity_scaler = StandardScaler()
quantity_scaler.fit(X_train['quantity'].values.reshape(-1,1)) # finding the me
an and standard deviation of this data
# print(f"Mean : {quantity_scaler.mean_[0]}, Standard deviation : {np.sqrt(qua
ntity_scaler.var_[0])}")

# Now standardize the data with above maen and variance.
quantity_standardized_train = quantity_scaler.transform(X_train['quantity'].va
lues.reshape(-1, 1))
# quantity_standardized_cv = quantity_scaler.transform(X_cv['quantity'].value
s.reshape(-1, 1))
quantity_standardized_test = quantity_scaler.transform(X_test['quantity'].valu
es.reshape(-1, 1))
print(quantity_standardized_train.shape)
# print(quantity_standardized_cv.shape)
print(quantity_standardized_test.shape)
```

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

```
(73196, 1)
(36052, 1)
```

```
In [36]: # check this one: https://www.youtube.com/watch?v=0H0q0cLn3Z4&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=[725.05 213.03 32
9. ... 399. 287.73 5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

# Now standardize the data with above mean and variance.
teacher_number_of_previously_posted_projects_standardized_train = teacher_number_of_previously_posted_projects_scalar.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
# teacher_number_of_previously_posted_projects_standardized_cv = teacher_number_of_previously_posted_projects_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
teacher_number_of_previously_posted_projects_standardized_test = teacher_number_of_previously_posted_projects_scalar.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
print(teacher_number_of_previously_posted_projects_standardized_train.shape)
# print(teacher_number_of_previously_posted_projects_standardized_cv.shape)
print(teacher_number_of_previously_posted_projects_standardized_test.shape)
```

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

```
(73196, 1)
(36052, 1)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [37]: X_train.head(2)

Out[37]:

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	prc
38506	80517 p041220	b8dd75437c64c51a21d980c87e86e05a	Mrs.	CA	
5861	133990 p058177	25e704869a2a1cdc944e6f83bbd56d5d	Mrs.	NJ	

Bag of Words(BOW) on project_TEXT/ESSAYS (Train,Cv,Test)

```
In [38]: # We are considering only the words which appeared in at least 10 documents(ro
ws or projects).
vectorizer_bow_essays = CountVectorizer(min_df=10,max_features=5000,ngram_range=(1,2))
vectorizer_bow_essays.fit(X_train['preprocessed_essays'])

text_bow_train = vectorizer_bow_essays.transform(X_train['preprocessed_essays'
])
# text_bow_cv = vectorizer_bow_essays.transform(X_cv['preprocessed_essays'])
text_bow_test = vectorizer_bow_essays.transform(X_test['preprocessed_essays'])
print("Shape of matrix after BOW_text_train ",text_bow_train.shape)
# print("Shape of matrix after BOW_text_cv ",text_bow_cv.shape)
print("Shape of matrix after BOW_text_test ",text_bow_test.shape)
```

Shape of matrix after BOW_text_train (73196, 5000)

Shape of matrix after BOW_text_test (36052, 5000)

Bag of Words(BOW) on project_title (Train,Cv,Test)

```
In [39]: # We are considering only the words which appeared in at least 10 documents(ros or projects).
vectorizer_bow_titles = CountVectorizer(min_df=10)
vectorizer_bow_titles.fit(X_train['preprocessed_titles'])

title_bow_train = vectorizer_bow_titles.transform(X_train['preprocessed_titles'])
# title_bow_cv = vectorizer_bow_titles.transform(X_cv['preprocessed_titles'])
title_bow_test = vectorizer_bow_titles.transform(X_test['preprocessed_titles'])
print("Shape of matrix after BOW_title_train ",title_bow_train.shape)
# print("Shape of matrix after BOW_title_cv ",title_bow_cv.shape)
print("Shape of matrix after BOW_title_test ",title_bow_test.shape)
```

Shape of matrix after BOW_title_train (73196, 2639)

Shape of matrix after BOW_title_test (36052, 2639)

TFIDF Vectorizer on project_TEXT/ESSAYS (Train,Cv,Test)

```
In [40]: from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essays = TfidfVectorizer(min_df=10,max_features=5000,ngram_range=(1,2))
vectorizer_tfidf_essays.fit(X_train['preprocessed_essays'])

text_tfidf_train = vectorizer_tfidf_essays.transform(X_train['preprocessed_essays'])
# text_tfidf_cv = vectorizer_tfidf_essays.transform(X_cv['preprocessed_essays'])
text_tfidf_test = vectorizer_tfidf_essays.transform(X_test['preprocessed_essays'])
print("Shape of matrix after tfidf_text_train ",text_tfidf_train.shape)
# print("Shape of matrix after tfidf_text_cv ",text_tfidf_cv.shape)
print("Shape of matrix after tfidf_text_test ",text_tfidf_test.shape)
```

Shape of matrix after tfidf_text_train (73196, 5000)

Shape of matrix after tfidf_text_test (36052, 5000)

TFIDF Vectorizer on project_title (Train,Cv,Test)

```
In [41]: from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(X_train['preprocessed_titles'])

title_tfidf_train = vectorizer_tfidf_title.transform(X_train['preprocessed_titles'])
# title_tfidf_cv = vectorizer_tfidf_title.transform(X_cv['preprocessed_titles'])
title_tfidf_test = vectorizer_tfidf_title.transform(X_test['preprocessed_titles'])
print("Shape of matrix after tfidf_title_train ",title_tfidf_train.shape)
# print("Shape of matrix after tfidf_title_cv ",title_tfidf_cv.shape)
print("Shape of matrix after tfidf_title_test ",title_tfidf_test.shape)
```

Shape of matrix after tfidf_title_train (73196, 2639)

Shape of matrix after tfidf_title_test (36052, 2639)

```
In [42]: # 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())
```

Avg W2V on TEXT/ESSAYS(Train,cv,test)


```

In [43]: # average Word2Vec
# compute average word2vec for each review.
avg_w2v_essays_vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm1(X_train['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_essays_vectors_train.append(vector)

# avg_w2v_essays_vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
# for sentence in tqdm1(X_cv['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_essays_vectors_cv.append(vector)

avg_w2v_essays_vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm1(X_test['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_essays_vectors_test.append(vector)

print(len(avg_w2v_essays_vectors_train))
# print(len(avg_w2v_essays_vectors_cv))
print(len(avg_w2v_essays_vectors_test))
print(len(avg_w2v_essays_vectors_train[0]))
# print(len(avg_w2v_essays_vectors_cv[0]))
print(len(avg_w2v_essays_vectors_test[0]))

```

73196
36052
300
300

Avg W2V on TITLES(Train,cv,test)

```

In [44]: # average Word2Vec
# compute average word2vec for each review.
avg_w2v_titles_vectors_train = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm1(X_train['preprocessed_titles']): # for each review/sente
nce
    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_titles_vectors_train.append(vector)

# avg_w2v_titles_vectors_cv = []; # the avg-w2v for each sentence/review is st
ored in this list
# for sentence in tqdm1(X_cv['preprocessed_titles']): # for each review/senten
ce
#     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_titles_vectors_cv.append(vector)

avg_w2v_titles_vectors_test = []; # the avg-w2v for each sentence/review is st
ored in this list
for sentence in tqdm1(X_test['preprocessed_titles']): # for each review/senten
ce
    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_titles_vectors_test.append(vector)

print(len(avg_w2v_titles_vectors_train))
# print(len(avg_w2v_titles_vectors_cv))
print(len(avg_w2v_titles_vectors_test))
print(len(avg_w2v_titles_vectors_train[0]))
# print(len(avg_w2v_titles_vectors_cv[0]))
print(len(avg_w2v_titles_vectors_test[0]))

```

73196
36052
300
300

TFIDF weighted W2V on TEXT/ESSAYS(Train,cv,test)

```

In [45]: # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['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())
*****
*
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_essays_vectors_train = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm1(X_train['preprocessed_essays']): # for each review/sente
nce
    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/revie
w
    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 v
alue((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_essays_vectors_train.append(vector)

# # average Word2Vec
# # compute average word2vec for each review.
# tfidf_w2v_essays_vectors_cv = []; # the avg-w2v for each sentence/review is
stored in this list
# for sentence in tqdm1(X_cv['preprocessed_essays']): # for each review/senten
ce
#     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/rev
iew
#     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.spl
it())) # 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_essays_vectors_cv.append(vector)

# average Word2Vec

```

```

# compute average word2vec for each review.
tfidf_w2v_essays_vectors_test = []; # the avg-w2v for each sentence/review is
    stored in this list
for sentence in tqdm1(X_test['preprocessed_essays']): # for each review/senten
    ce
        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/revie
        w
        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 v
                alue((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_essays_vectors_test.append(vector)

print(len(tfidf_w2v_essays_vectors_train))
# print(len(tfidf_w2v_essays_vectors_cv))
print(len(tfidf_w2v_essays_vectors_test))
print(len(tfidf_w2v_essays_vectors_train[0]))
# print(len(tfidf_w2v_essays_vectors_cv[0]))
print(len(tfidf_w2v_essays_vectors_test[0]))

```

```

73196
36052
300
300

```

TFIDF weighted W2V on TITLES(Train,cv,test)

```

In [46]: # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_titles'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())
*****
*
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_titles_vectors_train = []; # the avg-w2v for each sentence/review is
stored in this list
for sentence in tqdm1(X_train['preprocessed_titles']): # for each review/sente
nce
    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/revie
w
    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 v
alue((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_titles_vectors_train.append(vector)

# # average Word2Vec
# # compute average word2vec for each review.
# tfidf_w2v_titles_vectors_cv = []; # the avg-w2v for each sentence/review is
stored in this list
# for sentence in tqdm1(X_cv['preprocessed_titles']): # for each review/senten
ce
#     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/rev
iew
#     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.spl
it())) # 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_titles_vectors_cv.append(vector)

# average Word2Vec

```

```

# compute average word2vec for each review.
tfidf_w2v_titles_vectors_test = []; # the avg-w2v for each sentence/review is
    stored in this list
for sentence in tqdm1(X_test['preprocessed_titles']): # for each review/senten
ce
    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/revie
w
    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 v
            alue((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_titles_vectors_test.append(vector)

print(len(tfidf_w2v_titles_vectors_train))
# print(len(tfidf_w2v_titles_vectors_cv))
print(len(tfidf_w2v_titles_vectors_test))
print(len(tfidf_w2v_titles_vectors_train[0]))
# print(len(tfidf_w2v_titles_vectors_cv[0]))
print(len(tfidf_w2v_titles_vectors_test[0]))

```

```

73196
36052
300
300

```

```

In [48]: import dill
        # dill.dump_session('notebook_env.db')
        # dill.load_session('notebook_env.db')

```

```

In [49]: project_data.columns

```

```

Out[49]: Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
               'project_submitted_datetime', 'project_title',
               'project_resource_summary',
               'teacher_number_of_previously_posted_projects', 'project_is_approved',
               'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantit
               y',
               'digits_in_summary', 'clean_project_grade_category',
               'preprocessed_essays', 'preprocessed_titles'],
              dtype='object')

```

```

In [ ]:

```



```
In [50]: # please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# make sure you featurize train and test data separatly

# when you plot any graph make sure you use
# a. Title, that describes your plot, this will be very helpful to the reader
# b. Legends if needed
# c. X-axis label
# d. Y-axis label
```

2.4 Applng Support Vector Machines on different kind of featurization as mentioned in the instructions

Apply Support Vector Machines 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

2.4.1 Applying SVM on BOW, SET 1

```
In [51]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_one_hot_train, teacher_prefix_one_hot_train
               , project_grade_category_one_hot_train, price_standardized_train,
               , teacher_number_of_previously_posted_projects_standardized_train, text_bow_train, title_bow_train)).tocsr()
# X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_one_hot_cv, teacher_prefix_one_hot_cv
#               , project_grade_category_one_hot_cv, price_standardized_cv, quantity_standardized_cv
#               , teacher_number_of_previously_posted_projects_standardized_cv, text_bow_cv, title_bow_cv)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_one_hot_test, teacher_prefix_one_hot_test
               , project_grade_category_one_hot_test, price_standardized_test, quantity_standardized_test
               , teacher_number_of_previously_posted_projects_standardized_test, text_bow_test, title_bow_test)).tocsr()

print("Final Data matrix on BOW")
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 on BOW

(73196, 7741) (73196,)

(36052, 7741) (36052,)

=====

GridSearchCV using Penalty = 'l2'

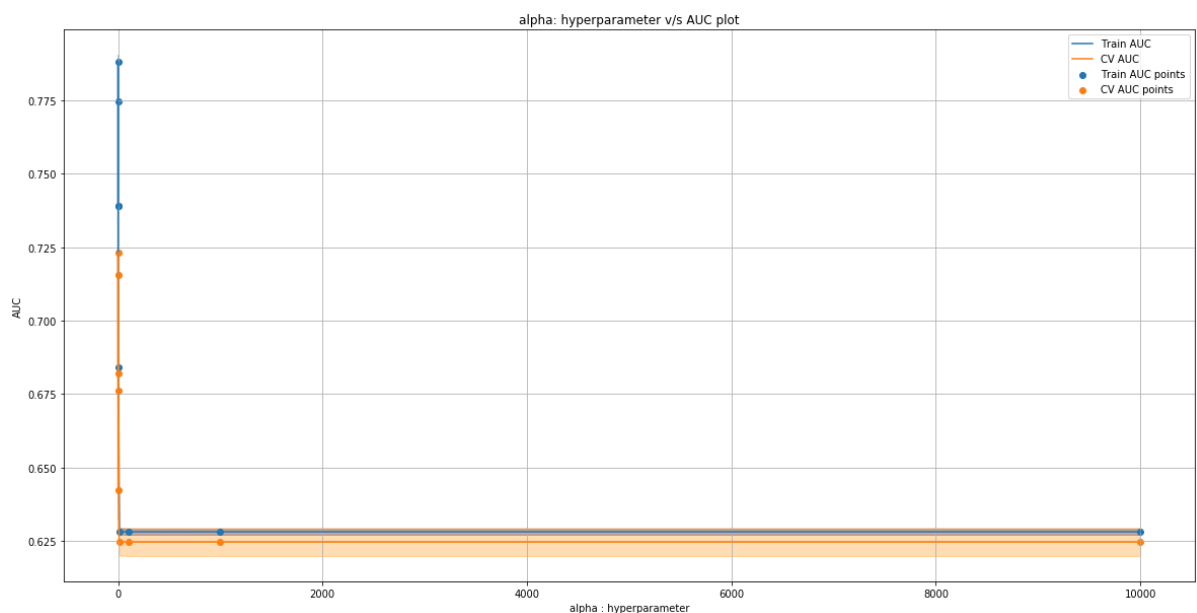
```
In [55]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV

sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
parameters = {'alpha': [10**-4, 10**-3, 10**-2, 10**-1, 10**0, 10**1, 10**2, 10**3, 10**4]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], train_auc - train_auc_std, train_auc +
train_auc_std, alpha=0.3, color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], cv_auc - cv_auc_std, cv_auc + cv_auc_std, alpha=0.3, color='darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()
```



Not able to visualise with high values of alphas, so we'll reduce

```

In [56]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV

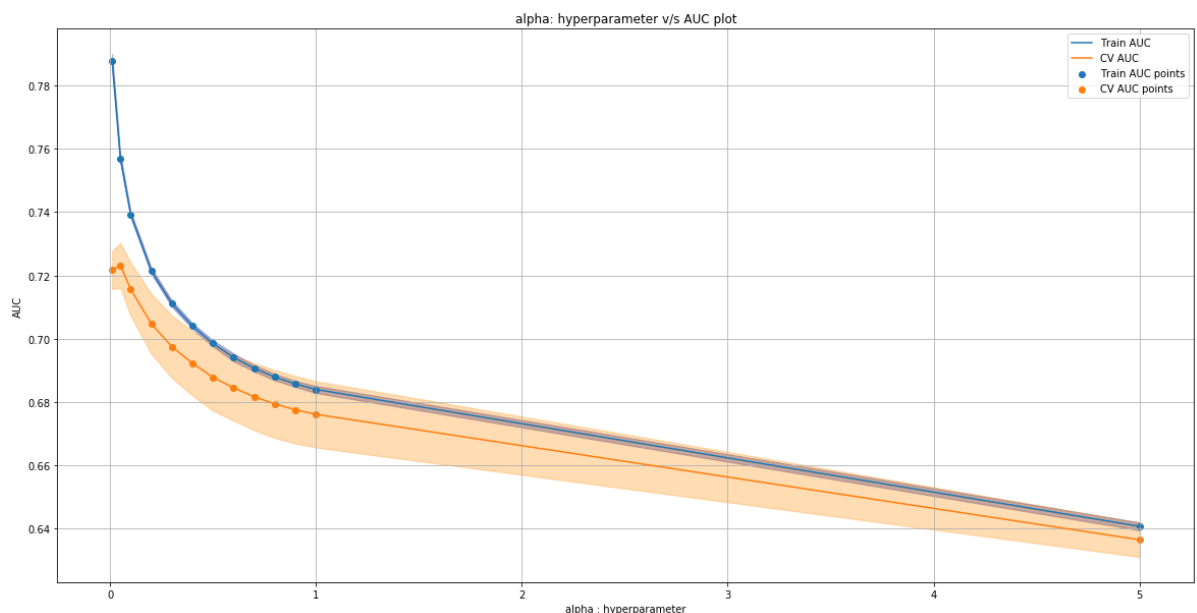
sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 5.0]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], train_auc - train_auc_std, train_auc +
+
train_auc_std, alpha=0.3, color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], cv_auc - cv_auc_std, cv_auc + cv_auc
_std, alpha=0.3, color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()

```



GridSearchCV using Penalty = 'l1'

```

In [73]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV

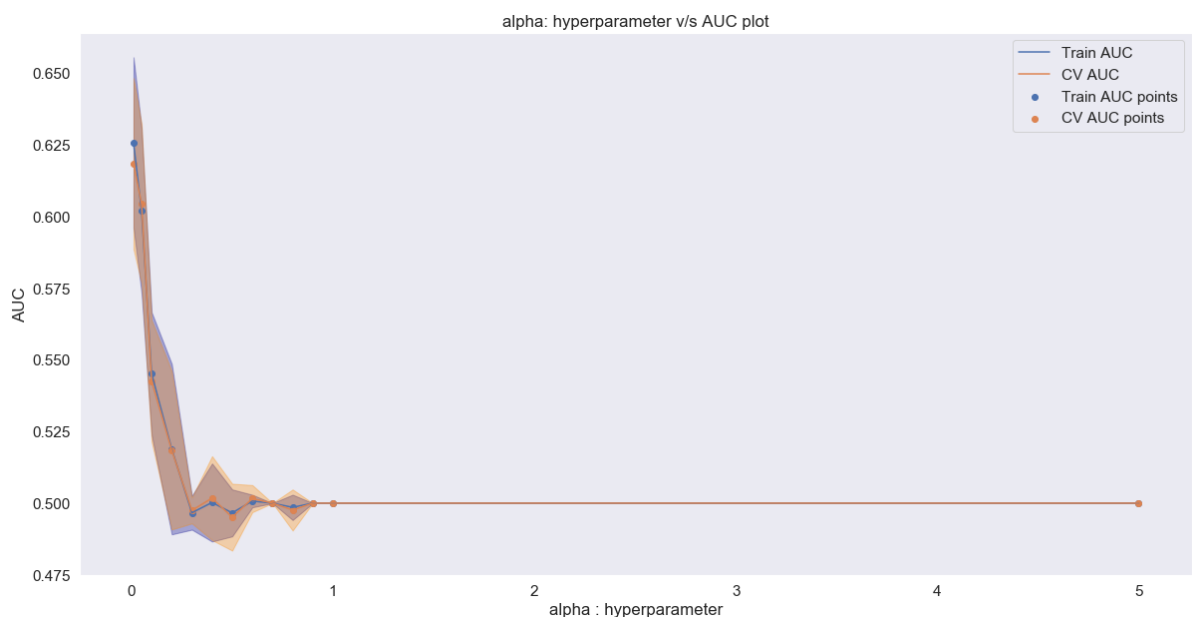
sv = SGDClassifier(loss='hinge', penalty='l1',class_weight='balanced')
parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
, 1.0, 5.0]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc
+
train_auc_std,alpha=0.3,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc
_std,alpha=0.3,color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()

```



L2 performs better

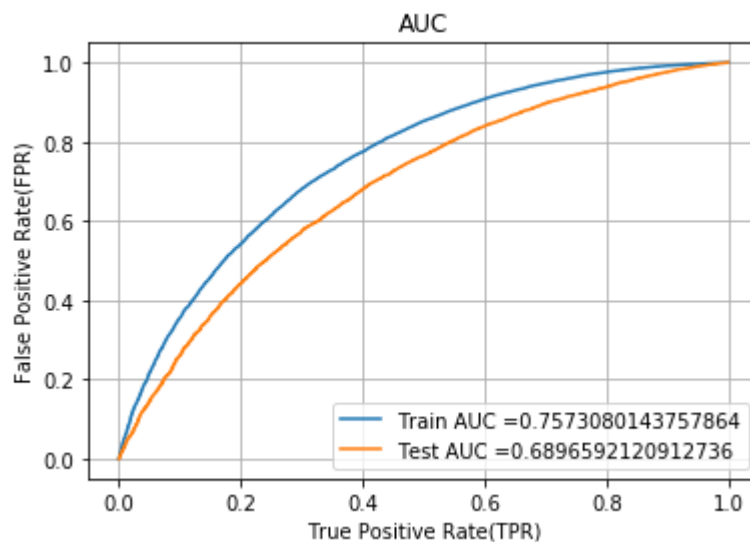
```
In [68]: from sklearn.metrics import roc_curve, auc

model = SGDClassifier(loss='hinge', penalty='l2', alpha=0.01)
model.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 = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)

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

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




```
In [69]: # 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 threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

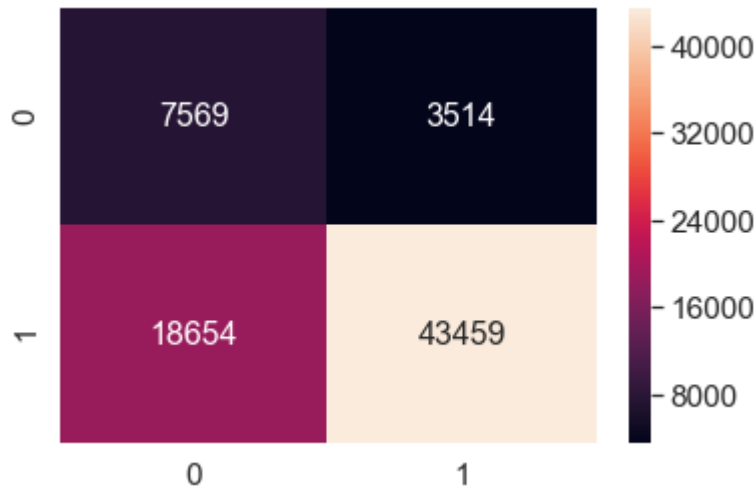
```
In [70]: print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:,], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:,], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.47783548165502 for threshold 1.091
[[ 7569  3514]
 [18654 43459]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.40881225069673194 for threshold 1.17
[[ 4149  1310]
 [15335 15258]]
```

```
In [71]: conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.47783548165502 for threshold 1.091

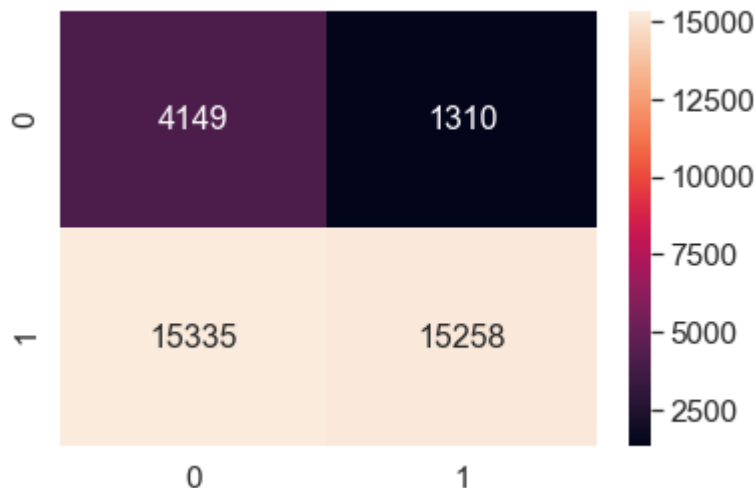
```
Out[71]: <matplotlib.axes._subplots.AxesSubplot at 0x1a5dea6bac8>
```



```
In [72]: conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.40881225069673194 for threshold 1.17

```
Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x1a5e0e3f0b8>
```



2.4.2 Applying SVM on TFIDF, SET 2

```
In [1]: import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress')
C:\Users\LENOVO\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

```
In [2]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_one_hot_train, teacher_prefix_one_hot_train,
               , project_grade_category_one_hot_train, price_standardized_train, quantity_standardized_train,
               , teacher_number_of_previously_posted_projects_standardized_train, text_tfidf_train, title_tfidf_train)).tocsr()
# X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_one_hot_cv, teacher_prefix_one_hot_cv,
#               , project_grade_category_one_hot_cv, price_standardized_cv, quantity_standardized_cv,
#               , teacher_number_of_previously_posted_projects_standardized_cv, text_tfidf_cv, title_tfidf_cv)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_one_hot_test, teacher_prefix_one_hot_test,
               , project_grade_category_one_hot_test, price_standardized_test, quantity_standardized_test,
               , teacher_number_of_previously_posted_projects_standardized_test, text_tfidf_test, title_tfidf_test)).tocsr()

print("Final Data matrix on TFIDF")
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 on TFIDF
```

```
(73196, 7741) (73196,)
```

```
(36052, 7741) (36052,)
```

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

GridSearchCV using Penalty = 'l2'

```

In [3]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
# https://machinelearningmastery.com/how-to-fix-futurewarning-messages-in-scik
it-learn/
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

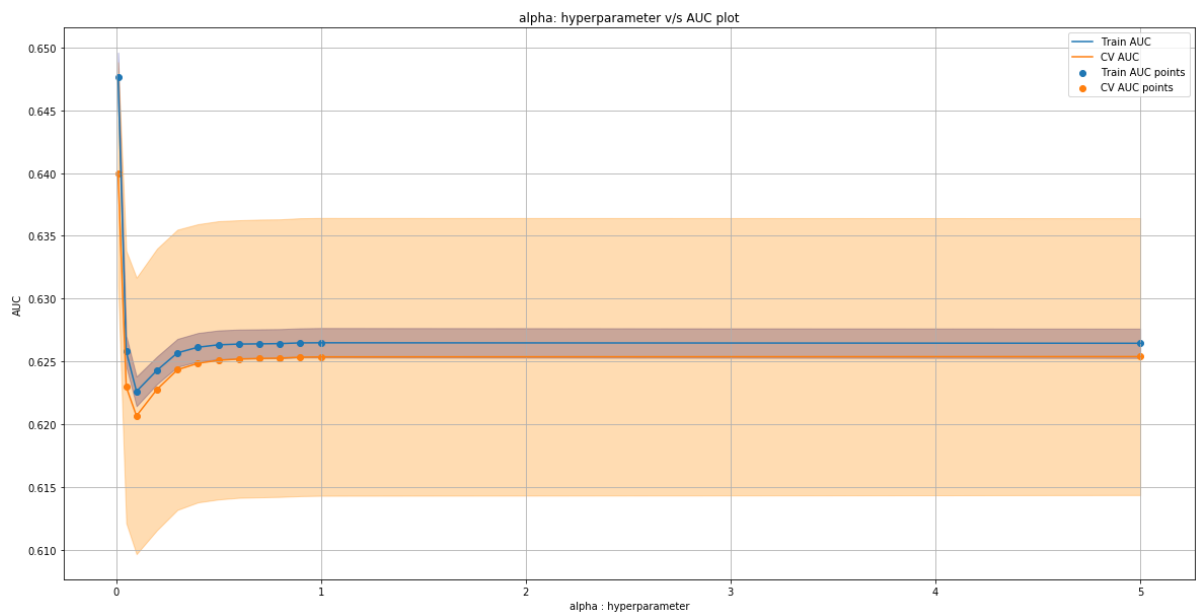
sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
, 1.0, 5.0]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], train_auc - train_auc_std, train_auc
+
train_auc_std, alpha=0.3, color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], cv_auc - cv_auc_std, cv_auc + cv_auc
_std, alpha=0.3, color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()

```



GridSearchCV using Penalty = 'l1'

```

In [4]: from sklearn.linear_model import SGDClassifier
        from sklearn.model_selection import GridSearchCV

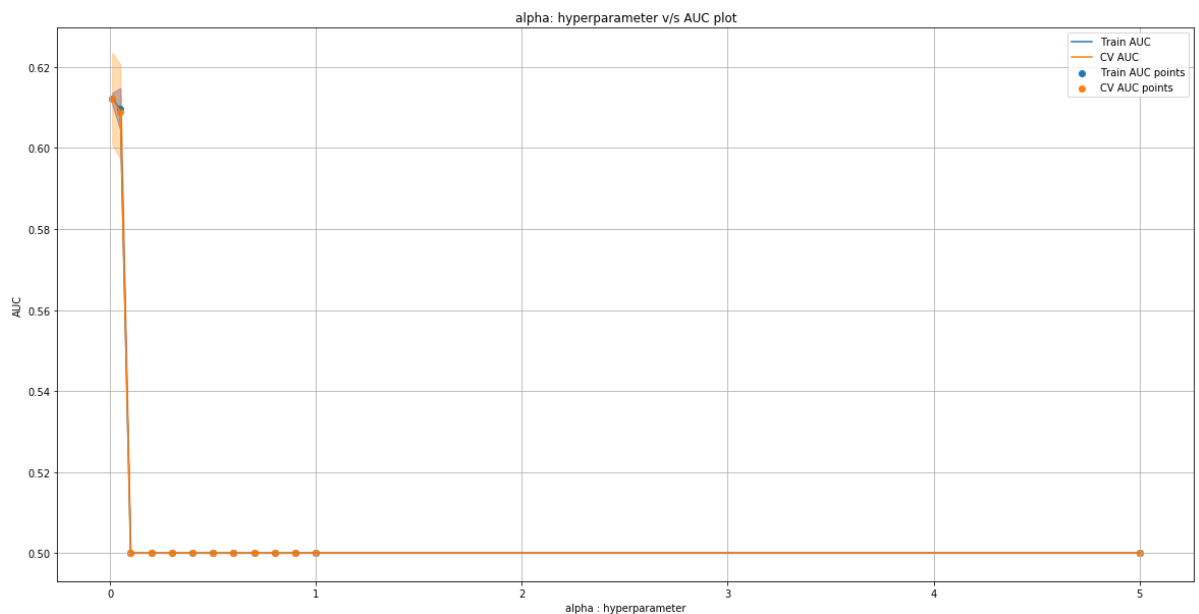
        sv = SGDClassifier(loss='hinge', penalty='l1', class_weight='balanced')
        parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
        , 1.0, 5.0]}

        clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
        plt.plot(parameters['alpha'], train_auc, label='Train AUC')
        # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
        plt.gca().fill_between(parameters['alpha'], train_auc - train_auc_std, train_auc
        +
        train_auc_std, alpha=0.3, color='darkblue')
        plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
        # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
        plt.gca().fill_between(parameters['alpha'], cv_auc - cv_auc_std, cv_auc + cv_auc
        _std, alpha=0.3, color=
        'darkorange')
        plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
        plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
        plt.legend()
        plt.xlabel("alpha : hyperparameter")
        plt.ylabel("AUC")
        plt.title("alpha: hyperparameter v/s AUC plot")
        plt.grid()
        plt.show()

```



Lets see for smaller values of alphas for L2

```
In [4]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV

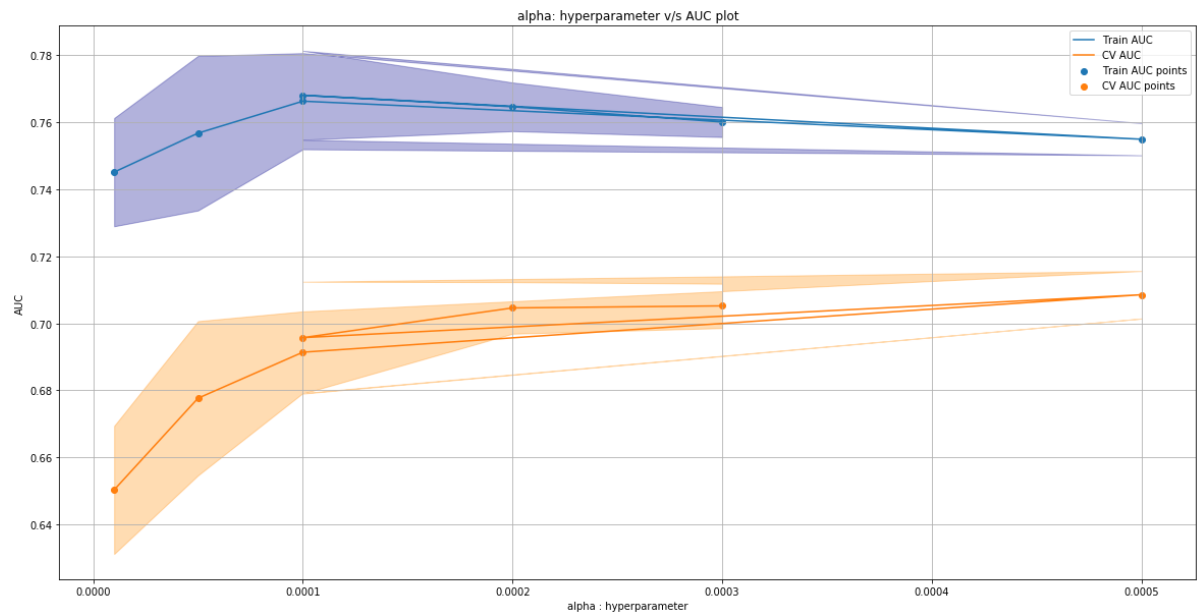
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
parameters = {'alpha':[0.00001,0.00005,0.0001, 0.0005, 0.001, 0.0002, 0.0003
]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc +
+
train_auc_std,alpha=0.3,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc
_std,alpha=0.3,color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()
```

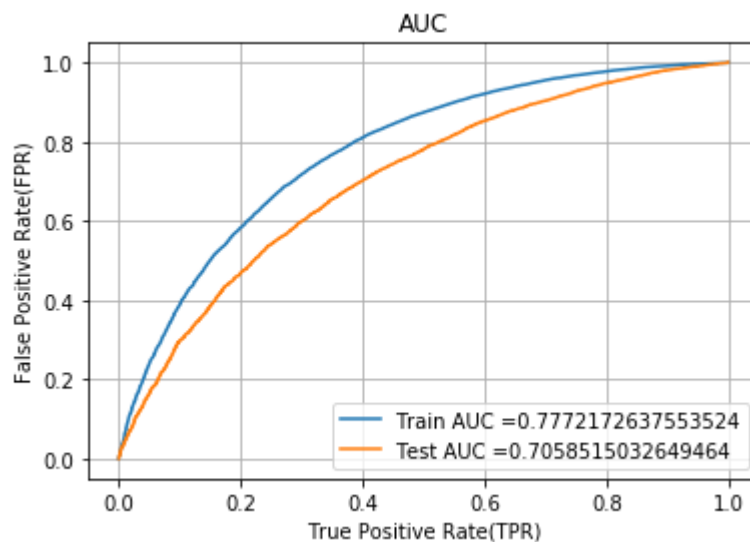
```
In [5]: from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import SGDClassifier

model = SGDClassifier(loss='hinge', penalty='l2', alpha=0.0002)
model.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 = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)

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

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



```
In [6]: # 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 threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

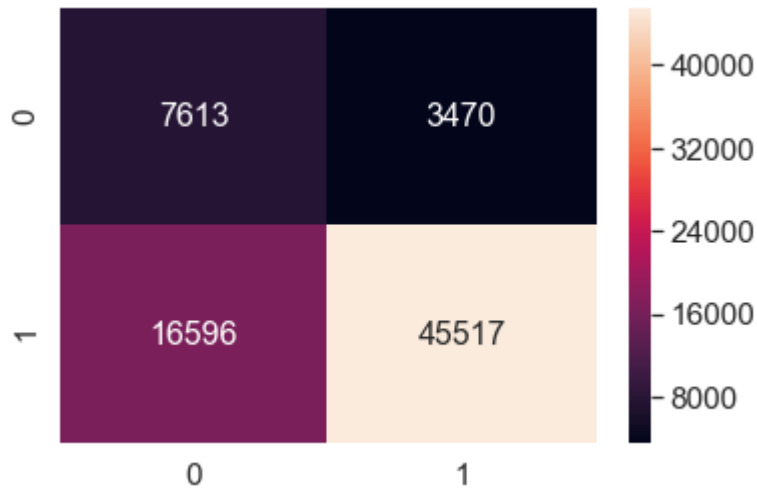
```
In [7]: print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:,], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:,], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5033726568377059 for threshold 1.084
[[ 7613  3470]
 [16596 45517]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4264309655128794 for threshold 1.173
[[ 4241  1218]
 [15214 15379]]
```

```
In [8]: conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.5033726568377059 for threshold 1.084

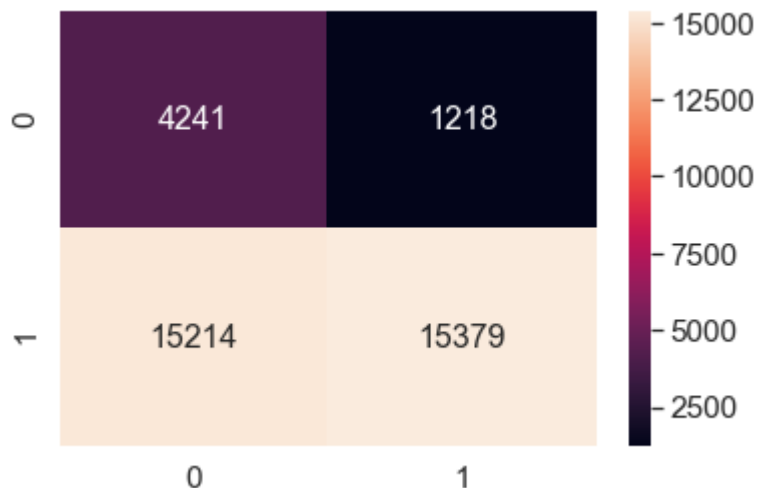
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1ddd6f833c8>



```
In [9]: conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.4264309655128794 for threshold 1.173

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1dde673ea58>



2.4.3 Applying SVM on AVG W2V, SET 3

```
In [1]: import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress')
C:\Users\LENOVO\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

```
In [2]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_one_hot_train, teacher_prefix_one_hot_train
              , project_grade_category_one_hot_train, price_standardized_train, quantity_standardized_train
              , teacher_number_of_previously_posted_projects_standardized_train, avg_w2v_essays_vectors_train
              , avg_w2v_titles_vectors_train)).tocsr()
# X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_one_hot_cv, teacher_prefix_one_hot_cv
#              , project_grade_category_one_hot_cv, price_standardized_cv, quantity_standardized_cv
#              , teacher_number_of_previously_posted_projects_standardized_cv, avg_w2v_essays_vectors_cv
#              , avg_w2v_titles_vectors_cv)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_one_hot_test, teacher_prefix_one_hot_test
              , project_grade_category_one_hot_test, price_standardized_test, quantity_standardized_test
              , teacher_number_of_previously_posted_projects_standardized_test, avg_w2v_essays_vectors_test
              , avg_w2v_titles_vectors_test)).tocsr()

print("Final Data matrix on AVGW2V")
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 on AVGW2V
```

```
(73196, 702) (73196,)
```

```
(36052, 702) (36052,)
```

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

GridSearchCV using Penalty = 'l2'

```

In [3]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
# https://machinelearningmastery.com/how-to-fix-futurewarning-messages-in-scikit-learn/
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

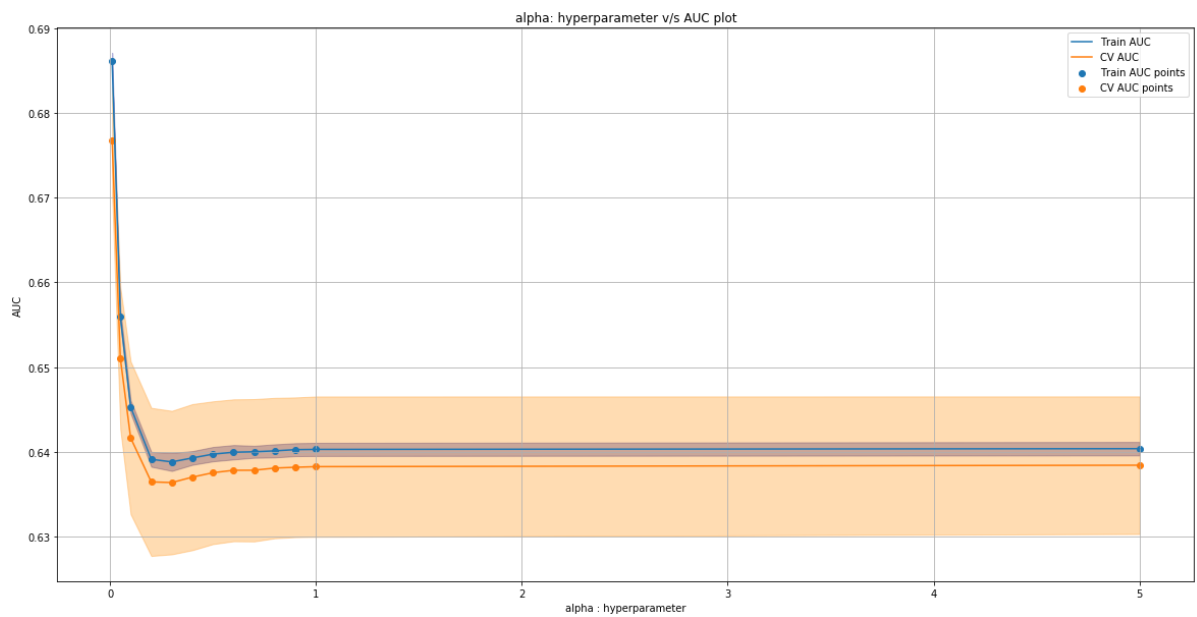
sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 5.0]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], train_auc - train_auc_std, train_auc +
+
train_auc_std, alpha=0.3, color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], cv_auc - cv_auc_std, cv_auc + cv_auc
_std, alpha=0.3, color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()

```



GridSearchCV using Penalty = 'l1'

```

In [4]: from sklearn.linear_model import SGDClassifier
        from sklearn.model_selection import GridSearchCV

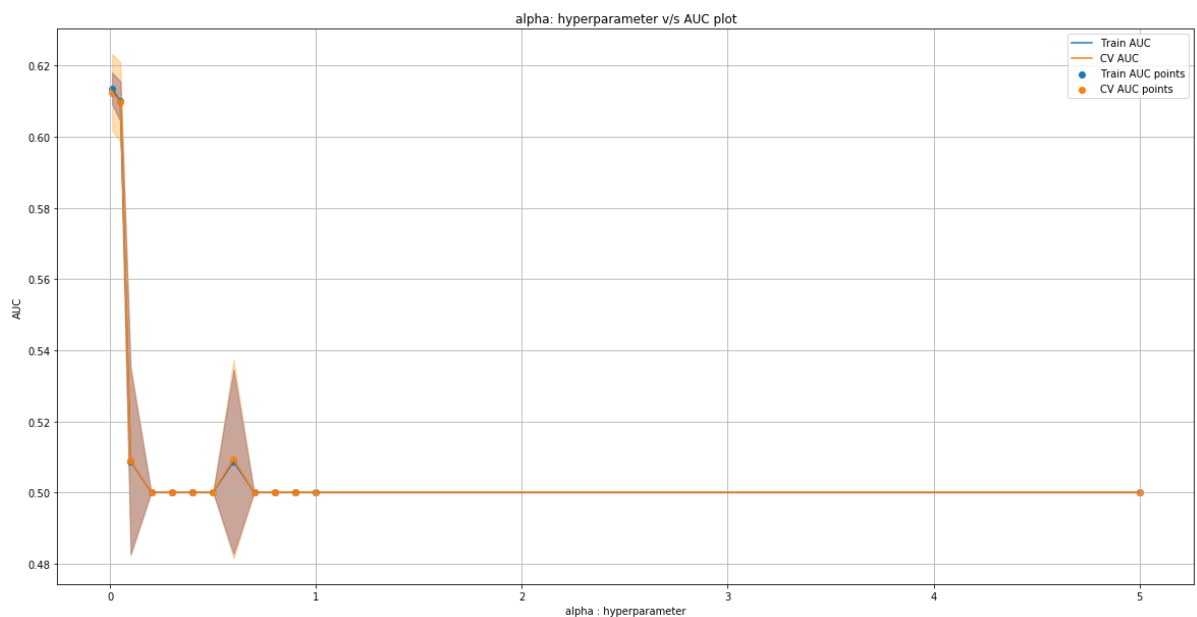
sv = SGDClassifier(loss='hinge', penalty='l1',class_weight='balanced')
parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
, 1.0, 5.0]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc
+
train_auc_std,alpha=0.3,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc
_std,alpha=0.3,color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()

```



Lets see for smaller values of alphas for L2

```
In [5]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV

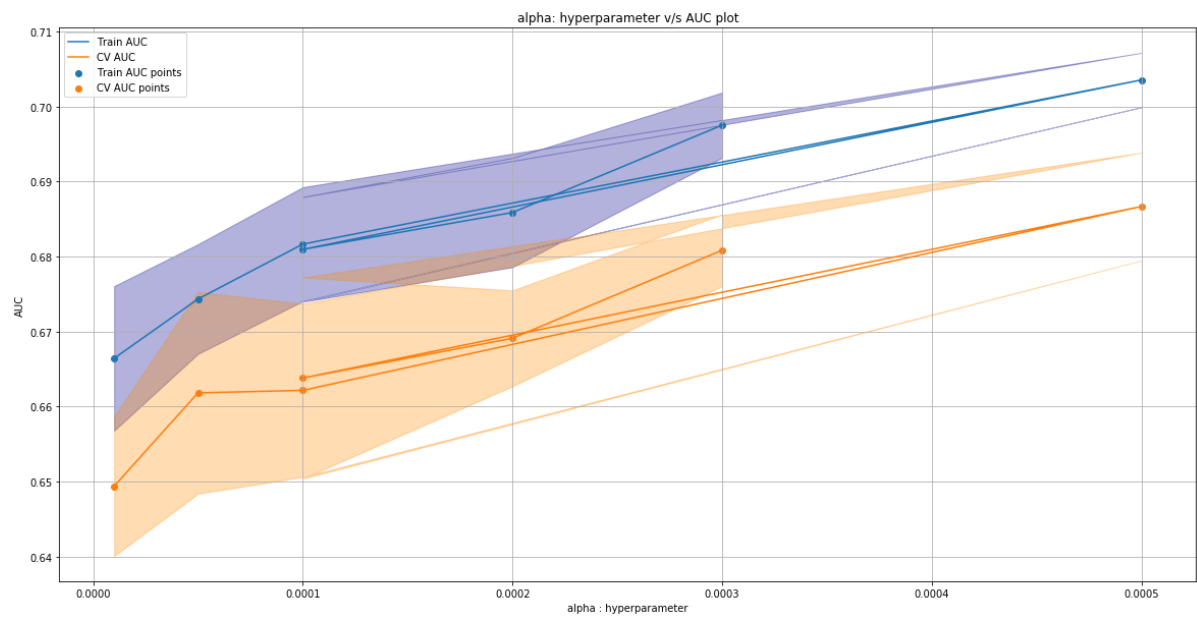
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
parameters = {'alpha':[0.00001,0.00005,0.0001, 0.0005, 0.001, 0.0002, 0.0003
]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc +
+
train_auc_std,alpha=0.3,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc
_std,alpha=0.3,color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()
```



```
In [3]: from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import SGDClassifier

model = SGDClassifier(loss='hinge', penalty='l2', alpha=0.0005)
model.fit(X_tr, y_train)

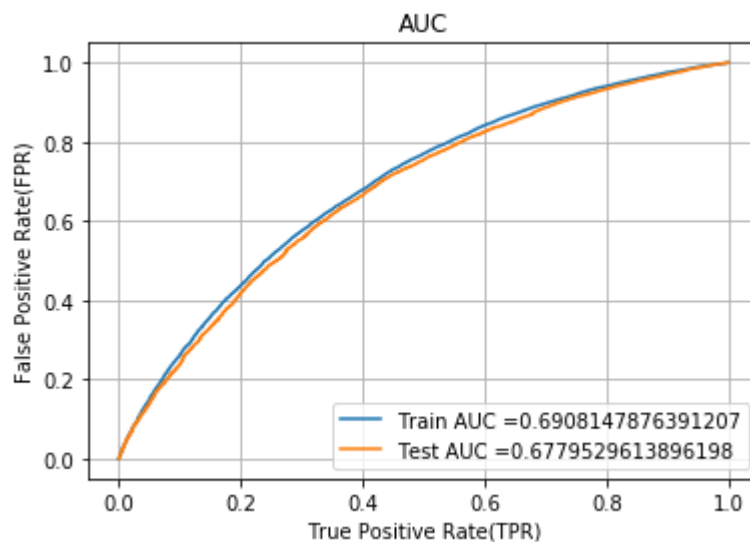
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)

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

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

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:

max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are left unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.



```
In [4]: # 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 threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

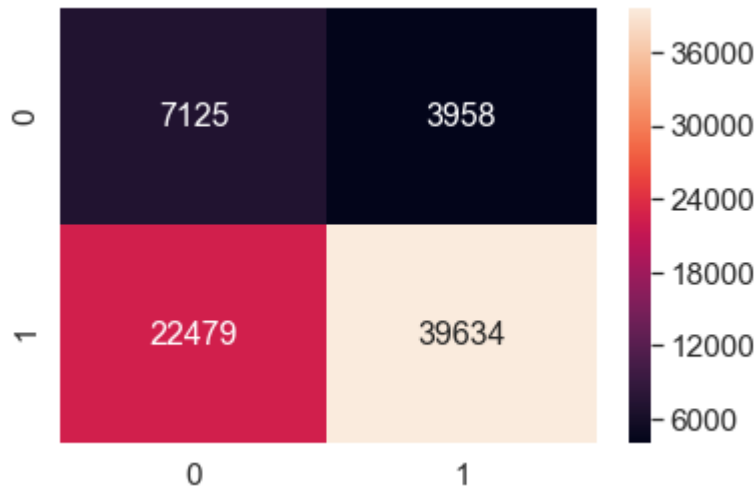
```
In [5]: print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:,], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:,], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

```
=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4102163203960711 for threshold 1.177
[[ 7125  3958]
 [22479 39634]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4027674808988909 for threshold 1.306
[[ 4415  1044]
 [18313 12280]]
```

```
In [6]: conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.4102163203960711 for threshold 1.177

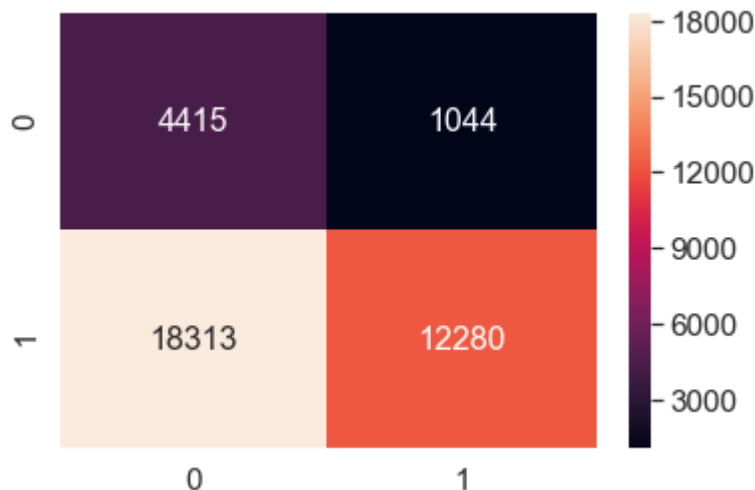
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x135e2583080>



```
In [7]: conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.4027674808988909 for threshold 1.306

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x135e272cf60>



2.4.4 Applying SVM on TFIDF W2V, SET 4

```
In [1]: import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress')
C:\Users\LENOVO\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

```
In [2]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_one_hot_train, teacher_prefix_one_hot_train
              , project_grade_category_one_hot_train, price_standardized_train, quantity_standardized_train
              , teacher_number_of_previously_posted_projects_standardized_train, tfidf_w2v_essays_vectors_train
              , tfidf_w2v_titles_vectors_train)).tocsr()
# X_cr = hstack((categories_one_hot_cv, sub_categories_one_hot_cv, school_state_one_hot_cv, teacher_prefix_one_hot_cv
#              , project_grade_category_one_hot_cv, price_standardized_cv, quantity_standardized_cv
#              , teacher_number_of_previously_posted_projects_standardized_cv, tfidf_w2v_essays_vectors_cv
#              , tfidf_w2v_titles_vectors_cv)).tocsr()
X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_one_hot_test, teacher_prefix_one_hot_test
              , project_grade_category_one_hot_test, price_standardized_test, quantity_standardized_test
              , teacher_number_of_previously_posted_projects_standardized_test, tfidf_w2v_essays_vectors_test
              , tfidf_w2v_titles_vectors_test)).tocsr()

print("Final Data matrix on TFIDF W2V")
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 on TFIDF W2V
```

```
(73196, 702) (73196,)
```

```
(36052, 702) (36052,)
```

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

GridSearchCV using Penalty = 'l2'

```

In [3]: from sklearn.linear_model import SGDClassifier
        from sklearn.model_selection import GridSearchCV
        # https://machinelearningmastery.com/how-to-fix-futurewarning-messages-in-scikit-learn/
        from warnings import simplefilter
        # ignore all future warnings
        simplefilter(action='ignore', category=FutureWarning)

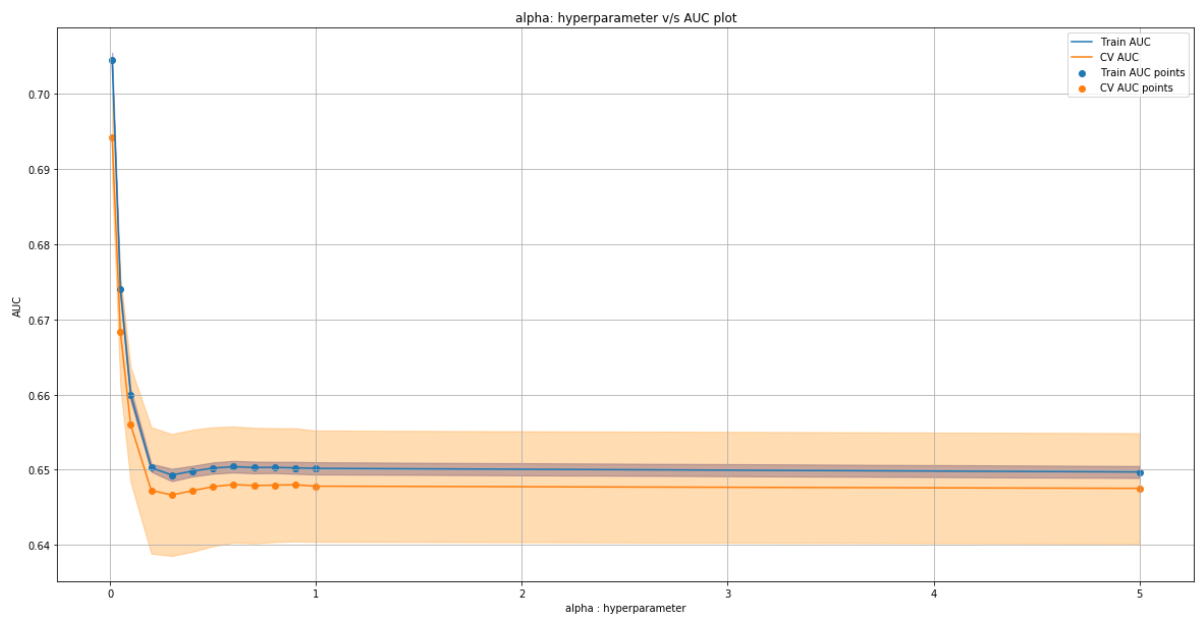
sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 5.0]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], train_auc - train_auc_std, train_auc +
+
train_auc_std, alpha=0.3, color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], cv_auc - cv_auc_std, cv_auc + cv_auc
_std, alpha=0.3, color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()

```

GridSearchCV using Penalty = 'l1'

```

In [4]: from sklearn.linear_model import SGDClassifier
        from sklearn.model_selection import GridSearchCV

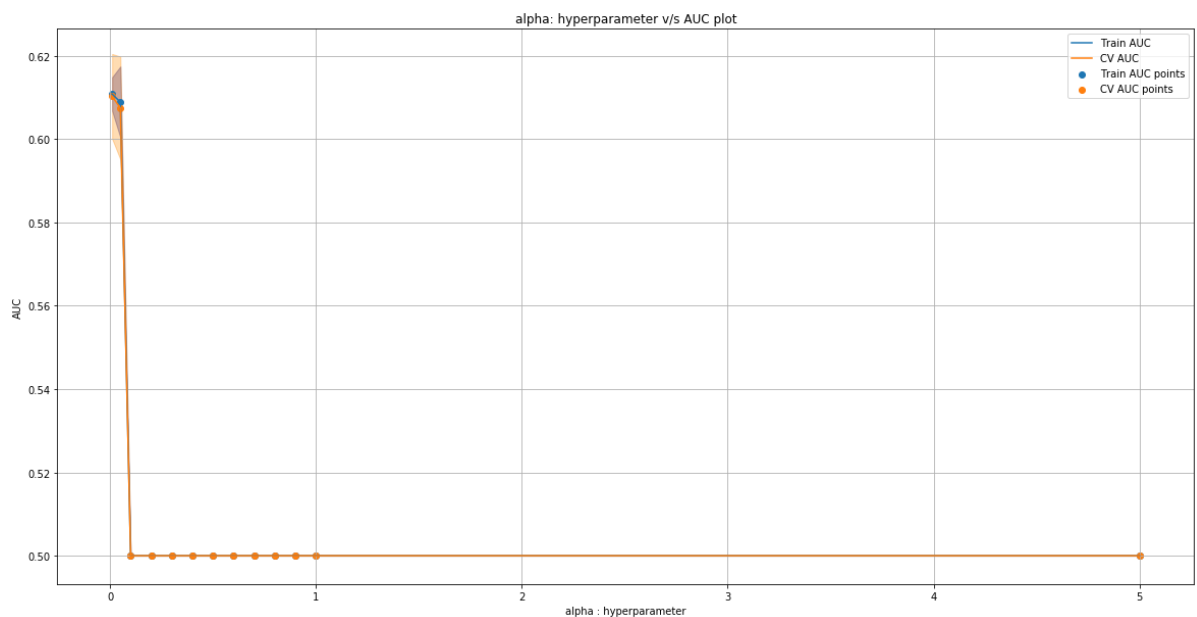
        sv = SGDClassifier(loss='hinge', penalty='l1', class_weight='balanced')
        parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
        , 1.0, 5.0]}

        clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
        plt.plot(parameters['alpha'], train_auc, label='Train AUC')
        # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
        plt.gca().fill_between(parameters['alpha'], train_auc - train_auc_std, train_auc
        +
        train_auc_std, alpha=0.3, color='darkblue')
        plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
        # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
        plt.gca().fill_between(parameters['alpha'], cv_auc - cv_auc_std, cv_auc + cv_auc
        _std, alpha=0.3, color=
        'darkorange')
        plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
        plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
        plt.legend()
        plt.xlabel("alpha : hyperparameter")
        plt.ylabel("AUC")
        plt.title("alpha: hyperparameter v/s AUC plot")
        plt.grid()
        plt.show()

```



Lets see for smaller values of alphas for L2

```

In [5]: from sklearn.linear_model import SGDClassifier
        from sklearn.model_selection import GridSearchCV

        from warnings import simplefilter
        # ignore all future warnings
        simplefilter(action='ignore', category=FutureWarning)

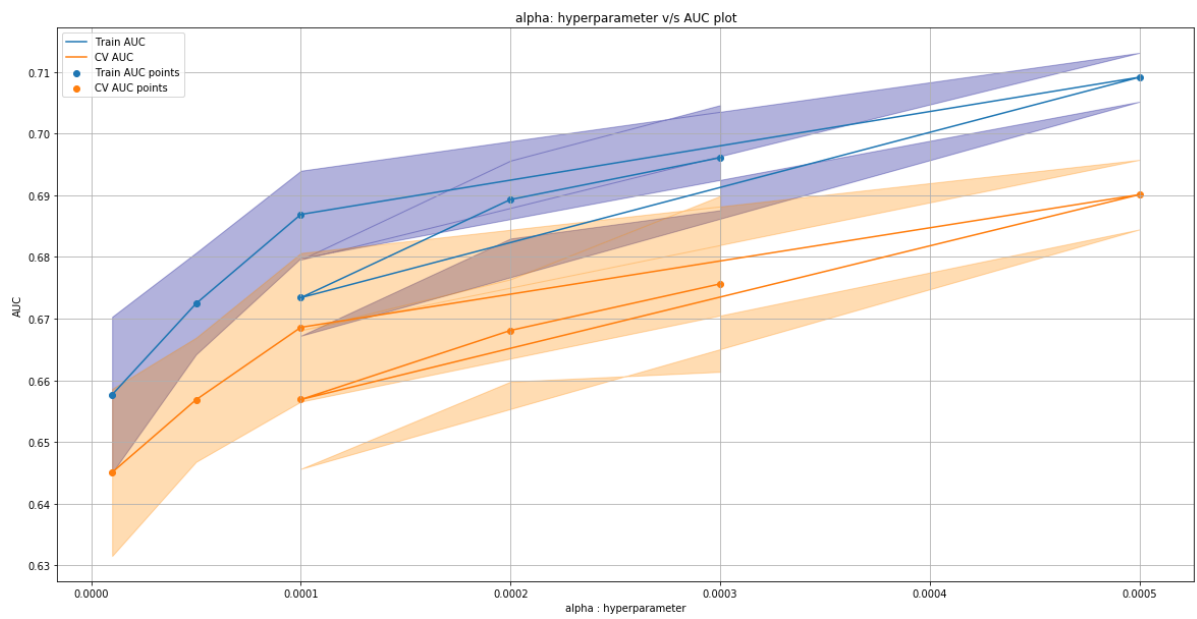
        sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
        parameters = {'alpha':[0.00001,0.00005,0.0001, 0.0005, 0.001, 0.0002, 0.0003
        ]}

        clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
        plt.plot(parameters['alpha'], train_auc, label='Train AUC')
        # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
        plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc
        +
        train_auc_std,alpha=0.3,color='darkblue')
        plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
        # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
        plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc
        _std,alpha=0.3,color=
        'darkorange')
        plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
        plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
        plt.legend()
        plt.xlabel("alpha : hyperparameter")
        plt.ylabel("AUC")
        plt.title("alpha: hyperparameter v/s AUC plot")
        plt.grid()
        plt.show()

```



```
In [3]: from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import SGDClassifier

model = SGDClassifier(loss='hinge', penalty='l2', alpha=0.0005)
model.fit(X_tr, y_train)

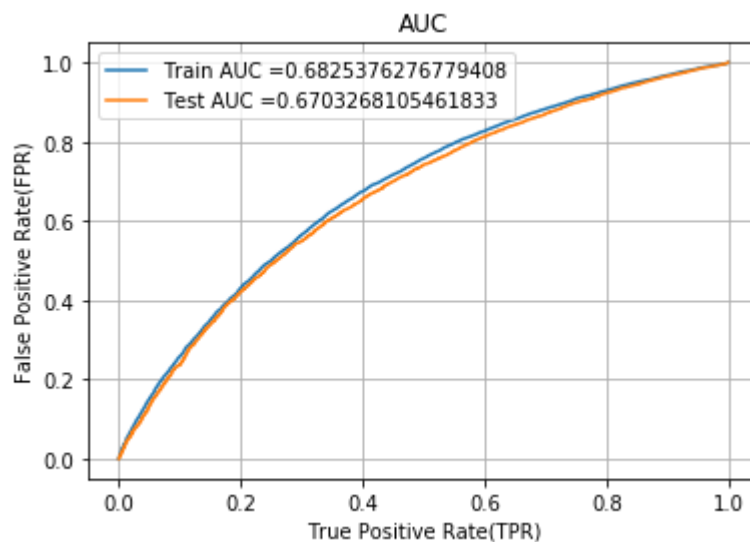
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive class
# not the predicted outputs
y_train_pred = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)

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

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

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:166: FutureWarning:

max_iter and tol parameters have been added in SGDClassifier in 0.19. If both are left unset, they default to max_iter=5 and tol=None. If tol is not None, max_iter defaults to max_iter=1000. From 0.21, default max_iter will be 1000, and default tol will be 1e-3.



```

In [4]: # 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 threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions

```

```

In [5]: print("="*100)
from sklearn.metrics import confusion_matrix
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_tpr)))

```

```

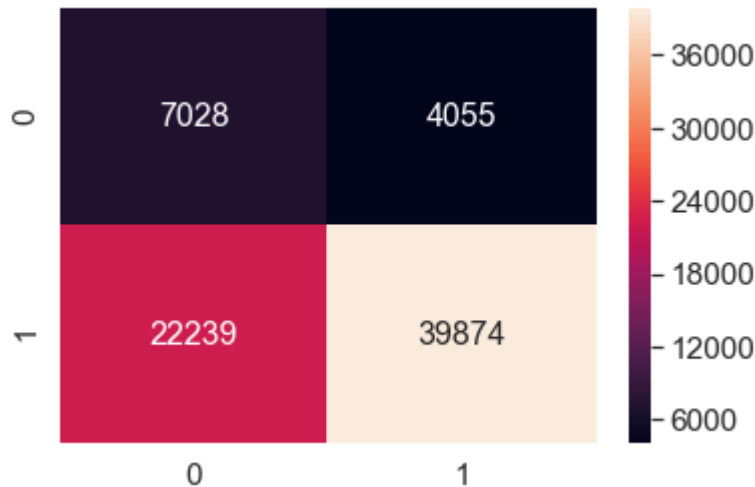
=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4070818301563723 for threshold 1.32
[[ 7028  4055]
 [22239 39874]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3957570999624106 for threshold 1.487
[[ 4474   985]
 [18553 12040]]

```

```
In [6]: conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.4070818301563723 for threshold 1.32

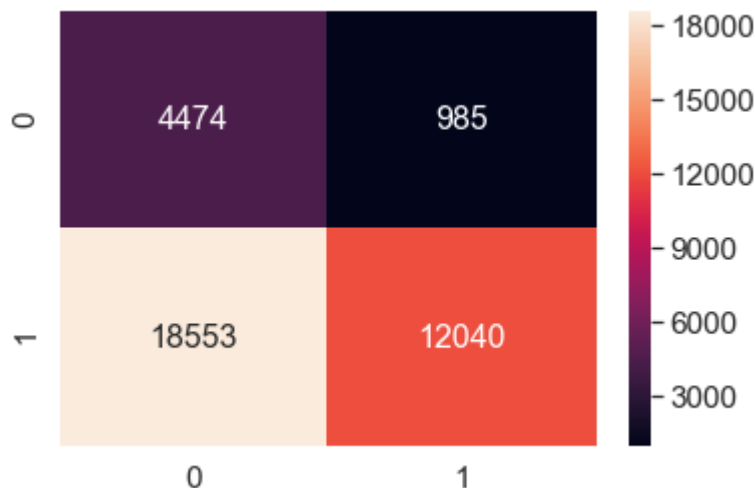
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x11fd29a5438>



```
In [7]: conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.3957570999624106 for threshold 1.487

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x11fd297aac8>



2.5 Support Vector Machines with added Features `Set 5`

2.5.1 Applying SVM on AVG W2V, SET 5

```
In [1]: import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\smart_open\ssh.py:34: UserWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled. `pip install paramiko` to suppress')
C:\Users\LENOVO\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

Adding number of words in TITLES feature

```
In [2]: print(X_train.shape)
print(X_test.shape)
```

```
(73196, 18)
(36052, 18)
```

```
In [3]: type(quantity_standardized_train)
```

```
Out[3]: numpy.ndarray
```

Word counts(TITLES)

```
In [14]: title_wordcount_train = []
title_train = list(X_train['preprocessed_titles'])
for i in tqdm1(title_train):
    b = len(str(i).split())
    title_wordcount_train.append(b)
title_wordcount_train = np.array(title_wordcount_train)

title_wordcount_test = []
title_test = list(X_test['preprocessed_titles'])
for i in tqdm1(title_test):
    b = len(str(i).split())
    title_wordcount_test.append(b)
title_wordcount_test = np.array(title_wordcount_test)

print(title_wordcount_train.shape)
print(title_wordcount_test.shape)
```

```
(73196,)
(36052,)
```

Standardizing Word counts(TITLES)

```
In [15]: from sklearn.preprocessing import StandardScaler

title_wordcount_scaler = StandardScaler()
title_wordcount_scaler.fit(title_wordcount_train.reshape(-1,1))

title_wordcount_standardized_train = title_wordcount_scaler.transform(title_wordcount_train.reshape(-1,1))
title_wordcount_standardized_test = title_wordcount_scaler.transform(title_wordcount_test.reshape(-1,1))

print(title_wordcount_standardized_train.shape)
print(title_wordcount_standardized_test.shape)
```

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int32 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int32 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int32 was converted to float64 by StandardScaler.

```
(73196, 1)
(36052, 1)
```

Word counts(ESSAYS)

```
In [16]: essay_wordcount_train = []
essay_train = list(X_train['preprocessed_essays'])
for i in tqdm1(essay_train):
    b = len(str(i).split())
    essay_wordcount_train.append(b)
essay_wordcount_train = np.array(essay_wordcount_train)

essay_wordcount_test = []
essay_test = list(X_test['preprocessed_titles'])
for i in tqdm1(essay_test):
    b = len(str(i).split())
    essay_wordcount_test.append(b)
essay_wordcount_test = np.array(essay_wordcount_test)

print(essay_wordcount_train.shape)
print(essay_wordcount_test.shape)
```

(73196,)

(36052,)

Standardizing Word counts(ESSAYS)

```
In [17]: from sklearn.preprocessing import StandardScaler

essay_wordcount_scalar = StandardScaler()
essay_wordcount_scalar.fit(essay_wordcount_train.reshape(-1,1))

essay_wordcount_standardized_train = essay_wordcount_scalar.transform(essay_wordcount_train.reshape(-1,1))
essay_wordcount_standardized_test = essay_wordcount_scalar.transform(essay_wordcount_test.reshape(-1,1))

print(essay_wordcount_standardized_train.shape)
print(essay_wordcount_standardized_test.shape)
```

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int32 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int32 was converted to float64 by StandardScaler.

C:\Users\LENOVO\Anaconda3\lib\site-packages\sklearn\utils\validation.py:595:
DataConversionWarning:

Data with input dtype int32 was converted to float64 by StandardScaler.

```
(73196, 1)
(36052, 1)
```

Sentiment scores for each essay

```

In [18]: import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

import nltk
# nltk.download('vader_Lexicon')

sid = SentimentIntensityAnalyzer()

essay_sentscore_train = []
essay_train = list(X_train['preprocessed_essays'])
for i in tqdm1(essay_train):
    ss = sid.polarity_scores(str(i))
    essay_sentscore_train.append(ss)
essay_sentscore_train = np.array(essay_sentscore_train)

essay_negscore_train = []
essay_neuscore_train = []
essay_posscore_train = []
essay_compoundscore_train = []
for it in essay_sentscore_train:
    a = it['neg']
    essay_negscore_train.append(a)
    b = it['neu']
    essay_neuscore_train.append(b)
    c = it['pos']
    essay_posscore_train.append(c)
    d = it['compound']
    essay_compoundscore_train.append(d)

essay_negscore_train = np.array(essay_negscore_train).reshape(-1,1)
essay_neuscore_train = np.array(essay_neuscore_train).reshape(-1,1)
essay_posscore_train = np.array(essay_posscore_train).reshape(-1,1)
essay_compoundscore_train = np.array(essay_compoundscore_train).reshape(-1,1)

print((essay_negscore_train.shape))
print((essay_neuscore_train.shape))
print((essay_posscore_train.shape))
print((essay_compoundscore_train.shape))

#####

essay_sentscore_test = []
essay_test = list(X_test['preprocessed_essays'])
for i in tqdm1(essay_test):
    ss = sid.polarity_scores(str(i))
    essay_sentscore_test.append(ss)
essay_sentscore_test = np.array(essay_sentscore_test)

essay_negscore_test = []
essay_neuscore_test = []
essay_posscore_test = []
essay_compoundscore_test = []
for it in essay_sentscore_test:
    a = it['neg']
    essay_negscore_test.append(a)

```

```

b = it['neu']
essay_neuscore_test.append(b)
c = it['pos']
essay_posscore_test.append(c)
d = it['compound']
essay_compoundscore_test.append(d)

essay_negscore_test = np.array(essay_negscore_test).reshape(-1,1)
essay_neuscore_test = np.array(essay_neuscore_test).reshape(-1,1)
essay_posscore_test = np.array(essay_posscore_test).reshape(-1,1)
essay_compoundscore_test = np.array(essay_compoundscore_test).reshape(-1,1)

print((essay_negscore_test.shape))
print((essay_neuscore_test.shape))
print((essay_posscore_test.shape))
print((essay_compoundscore_test.shape))

```

```

(73196, 1)
(73196, 1)
(73196, 1)
(73196, 1)

```

```

(36052, 1)
(36052, 1)
(36052, 1)
(36052, 1)

```

```

In [2]: from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essays = TfidfVectorizer(min_df=10,max_features=1010,ngram_range=(1,2))
vectorizer_tfidf_essays.fit(X_train['preprocessed_essays'])

text_tfidf_train1 = vectorizer_tfidf_essays.transform(X_train['preprocessed_essays'])
# text_tfidf_cv = vectorizer_tfidf_essays.transform(X_cv['preprocessed_essays'])
text_tfidf_test1 = vectorizer_tfidf_essays.transform(X_test['preprocessed_essays'])

```

```

In [3]: text_tfidf_train1.shape

```

```

Out[3]: (73196, 1010)

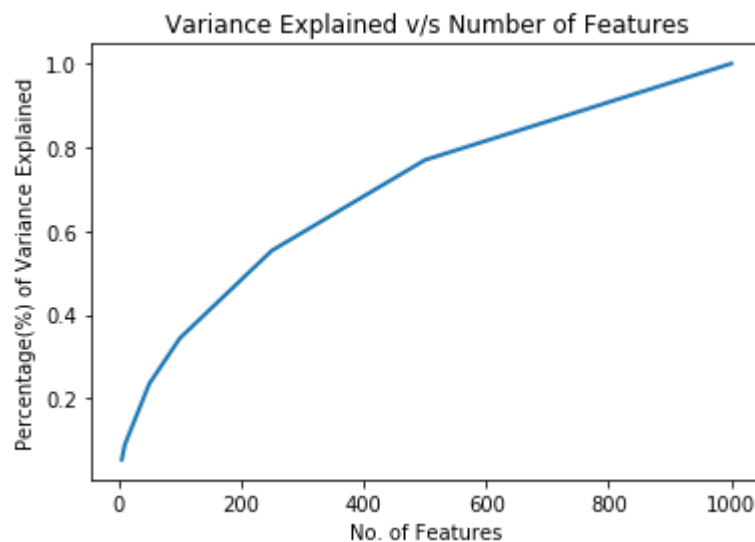
```

```
In [7]: from sklearn.decomposition import TruncatedSVD

features = [5,10,50,100,250,500,1000]
variance = []
# https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

for i in tqdm1(features):
    svd = TruncatedSVD(n_components= i, n_iter=10, random_state=1)
    svd.fit(text_tfidf_train1)
    variance.append(svd.explained_variance_ratio_.sum())
```

```
In [8]: plt.xlabel("No. of Features")
plt.ylabel("Percentage(%) of Variance Explained")
plt.title("Variance Explained v/s Number of Features")
plt.plot(features,variance,lw=2)
plt.show()
```



We'll pick the n-components as 1000 as it preserves above 95% of the variance

```
In [12]: svd = TruncatedSVD(n_components= 1000, n_iter=7, random_state=42)
svd.fit(text_tfidf_train)
svd_train = svd.transform(text_tfidf_train)
svd_test = svd.transform(text_tfidf_test)
```

```
In [13]: print(svd_train.shape)
print(svd_test.shape)
```

```
(73196, 1000)
(36052, 1000)
```



```
In [19]: # merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((categories_one_hot_train, sub_categories_one_hot_train, school_state_one_hot_train, teacher_prefix_one_hot_train
               , project_grade_category_one_hot_train, price_standardized_train,
               quantity_standardized_train
               , teacher_number_of_previously_posted_projects_standardized_train,
               title_wordcount_standardized_train
               , essay_wordcount_standardized_train, essay_negscore_train, essay_neuscore_train, essay_posscore_train
               , essay_compoundscore_train, svd_train)).tocsr()

X_te = hstack((categories_one_hot_test, sub_categories_one_hot_test, school_state_one_hot_test, teacher_prefix_one_hot_test
               , project_grade_category_one_hot_test, price_standardized_test, quantity_standardized_test
               , teacher_number_of_previously_posted_projects_standardized_test
               , title_wordcount_standardized_test
               , essay_wordcount_standardized_test, essay_negscore_test, essay_neuscore_test, essay_posscore_test
               , essay_compoundscore_test, svd_test)).tocsr()

print("Final Data matrix on TFIDF W2V")
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 on TFIDF W2V

(73196, 1108) (73196,)

(36052, 1108) (36052,)

=====

=====

GridSearchCV using Penalty = 'l2'

```

In [20]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV
# https://machinelearningmastery.com/how-to-fix-futurewarning-messages-in-scik
it-learn/
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

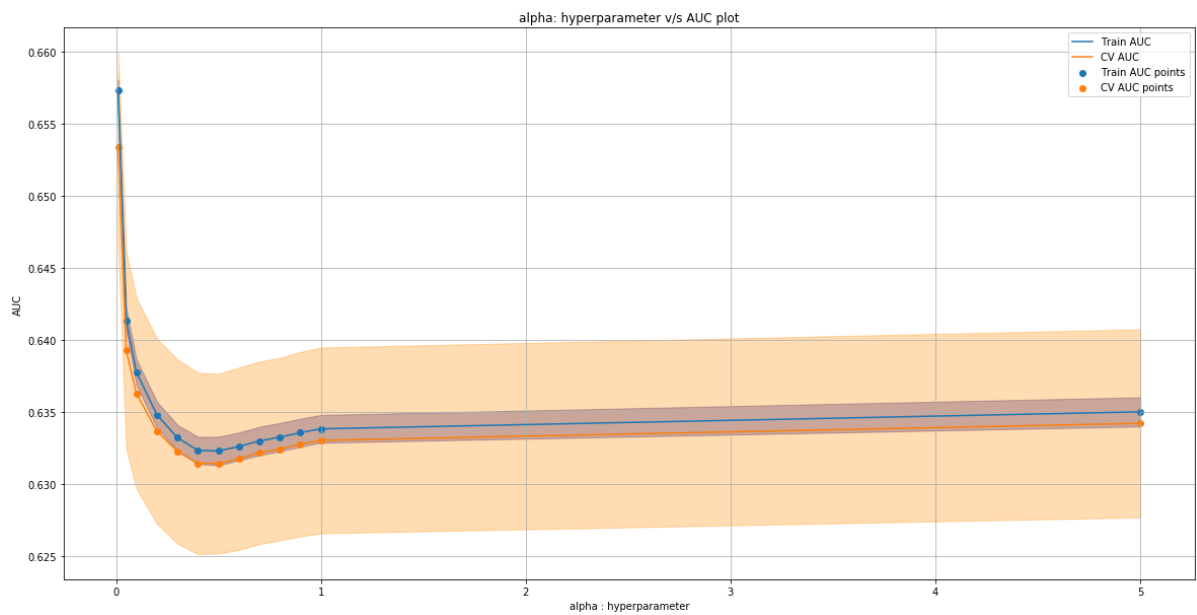
sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
, 1.0, 5.0]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], train_auc - train_auc_std, train_auc
+
train_auc_std, alpha=0.3, color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'], cv_auc - cv_auc_std, cv_auc + cv_auc
_std, alpha=0.3, color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()

```



GridSearchCV using Penalty = 'l1'

```

In [21]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV

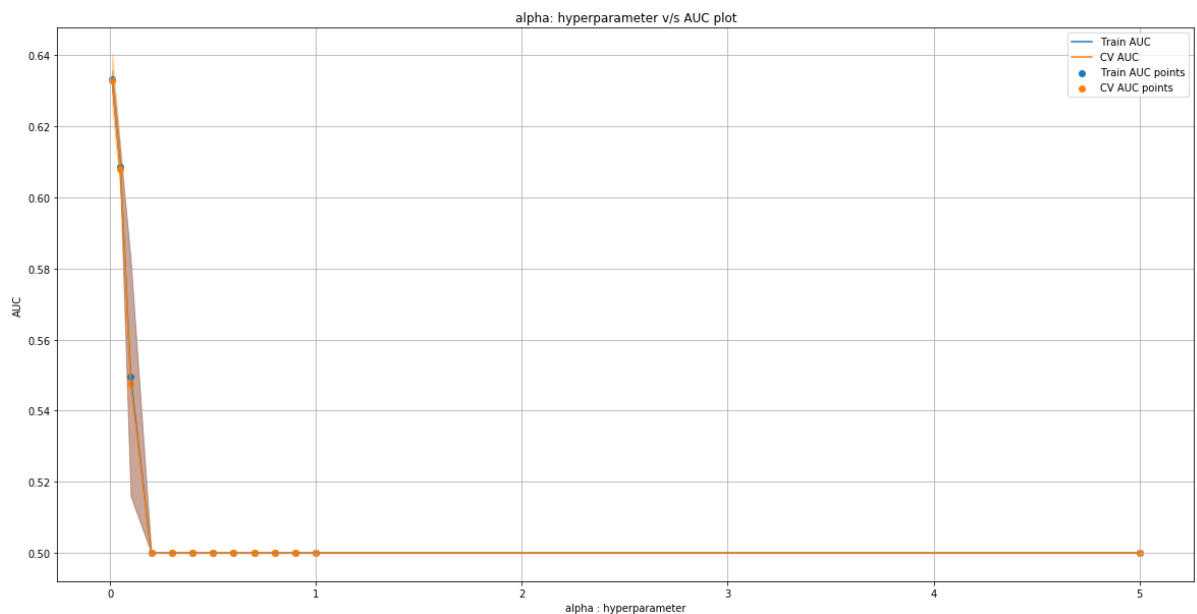
sv = SGDClassifier(loss='hinge', penalty='l1',class_weight='balanced')
parameters = {'alpha':[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
, 1.0, 5.0]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc
+
train_auc_std,alpha=0.3,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc
_std,alpha=0.3,color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()

```



Lets see for smaller values of alphas for L2

```
In [22]: from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV

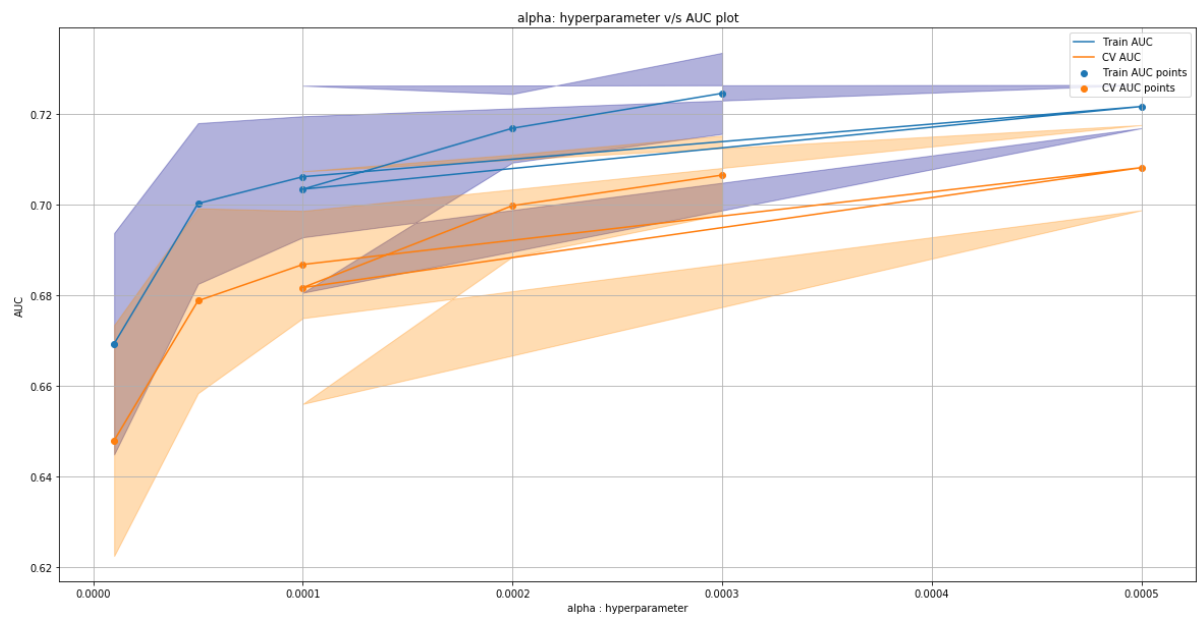
from warnings import simplefilter
# ignore all future warnings
simplefilter(action='ignore', category=FutureWarning)

sv = SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
parameters = {'alpha':[0.00001,0.00005,0.0001, 0.0005, 0.001, 0.0002, 0.0003
]}

clf = GridSearchCV(sv, 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.figure(figsize=(20,10))
plt.plot(parameters['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],train_auc - train_auc_std,train_auc +
+
train_auc_std,alpha=0.3,color='darkblue')
plt.plot(parameters['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(parameters['alpha'],cv_auc - cv_auc_std,cv_auc + cv_auc
_std,alpha=0.3,color=
'darkorange')
plt.scatter(parameters['alpha'], train_auc, label='Train AUC points')
plt.scatter(parameters['alpha'], cv_auc, label='CV AUC points')
plt.legend()
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("alpha: hyperparameter v/s AUC plot")
plt.grid()
plt.show()
```



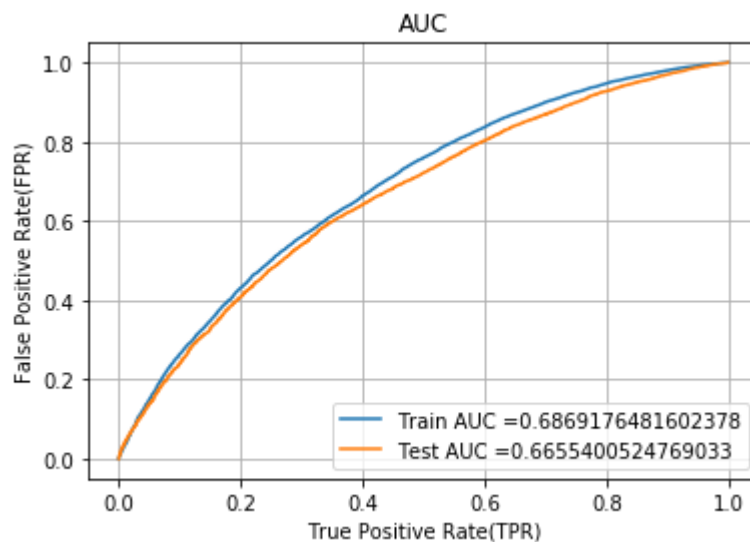
```
In [23]: from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import SGDClassifier

model = SGDClassifier(loss='hinge', penalty='l2', alpha=0.0005)
model.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 = model.decision_function(X_tr)
y_test_pred = model.decision_function(X_te)

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

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




```
In [24]: # 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 threshold", np.round(t,3))
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

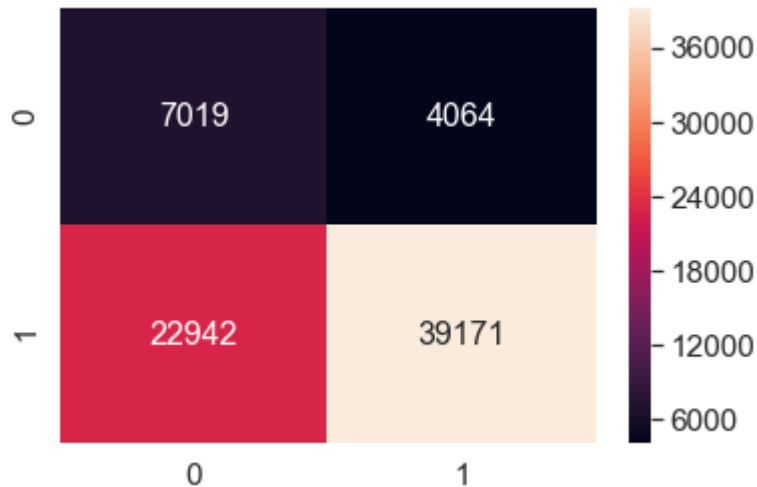
```
In [25]: print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:,], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:,], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))

=====
=====
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.39939264441527694 for threshold 1.044
[[ 7019  4064]
 [22942 39171]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.38964115957476736 for threshold 1.097
[[ 5394    65]
 [29192 1401]]
```

```
In [26]: conf_matr_df_train = pd.DataFrame(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.39939264441527694 for threshold 1.044

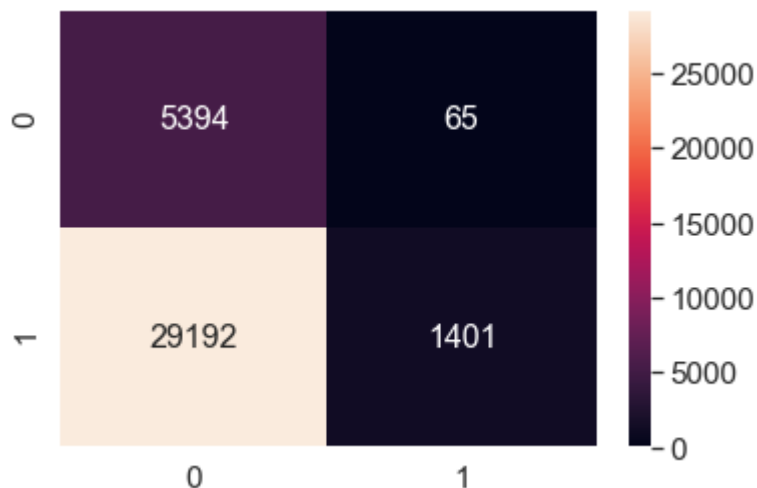
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1b5db3c2be0>



```
In [27]: conf_matr_df_test = pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

the maximum value of $tpr*(1-fpr)$ 0.38964115957476736 for threshold 1.097

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1b5d7b2c9b0>



```
In [0]: # please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
# a. Title, that describes your plot, this will be very helpful to the reader
# b. Legends if needed
# c. X-axis Label
# d. Y-axis Label
```

3. Conclusion

```
In [0]: # Please compare all your models using Prettytable library
```

```
In [28]: from prettytable import PrettyTable
#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]
x.add_row(["BOW", "SVM", 0.01, 0.68])
x.add_row(["TFIDF", "SVM", 0.0002, 0.70])
x.add_row(["AVG W2V", "SVM", 0.0005, 0.67])
x.add_row(["TFIDF W2V", "SVM", 0.0005, 0.67])
x.add_row(["TRUNCATED SVD", "SVM", 0.0005, 0.66])
print(x)
```

Vectorizer	Model	Hyper Parameter	AUC
BOW	SVM	0.01	0.68
TFIDF	SVM	0.0002	0.7
AVG W2V	SVM	0.0005	0.67
TFIDF W2V	SVM	0.0005	0.67
TRUNCATED SVD	SVM	0.0005	0.66

```
In [ ]:
```