Document Created Date:24-09-2019

DonorsChoose.org
Support a classroom. Build a future.

Understanding the Data and DataSource - DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

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

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

Support Vector Machine

```
Step by Step Procedure
```

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

- Vectorization ctegorical data (one hot encoding)
- Vectorizing text data(bow, tfidf, avgw2v, tfidf weighted w2v)
- vectorizing numarical - Normalizer
- Computing Sentiment Scores
- Applying Support Vector Machines
- Contactinating all the type of features(cat + text + num)
- Hyper parameter Tuning to find alpha:: Simple cross Validation (applied two techniques - this is one)
- Hyperparameter tuning to find th best estimator(RandomizedSearchCV- 2nd technique)
- Train the Support Vector Machines model using best hyperparameter and ploting auc roc-curve
- Ploting confusion matrix(heatmaps)
- Appling Support Vector Machines on different kind of featurization -Performing Elbow method to find the best number of Components
- Contactinating all the type of features(cat + text + num + svd + sentiment score feature)
- Hyper parameter Tuning to find alpha:: Simple cross Validation (applied two techniques - this is one)
- Hyperparameter tuning to find th best estimator(RandomizedSearchCV- 2nd technique)
- Train the Support Vector Machines model using best hyperparameter and ploting auc roc-curve
- Ploting confusion matrix(heatmaps)
- Observation on overall model performences (Conclusion)
- Ploting the performences by table format.

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

In [1]:

```
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [2]:

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

In [3]:

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

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

In [4]:

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


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


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


project_data.head(2)
```

Out[4]:

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

In [5]:

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

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

Out[5]:

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

1.1.1 preprocessing of `project_subject_categories`

In [6]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47

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

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

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

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

1.1.2 preprocessing of `project_subject_subcategories`

In [7]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47

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

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

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

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

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

1.1.3 preprocessing of school_state

In [8]:

```
school_state_catogories = list(project_data['school_state'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47

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

project_data['school_state'] = cat_list


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

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

1.1.4 preprocessing of teacher_prefix

In [9]:

```
# citation code :https://www.datacamp.com/community/tutorials/categorical-data
project_data = project_data.fillna(project_data['teacher_prefix'].value_counts().index[
teacher_prefix_catogories = list(project_data['teacher_prefix'].values)
# Citation code : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-sciki
# To convert the data type object to unicode string : used """astype('U')""" code from
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# remove special characters from list of strings python: https://stackoverflow.com/a/47
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-stri
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyth
cat_list = []
for i in teacher_prefix_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth
        if 'The' in j.split(): # this will split each of the catogory based on space "M
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"M
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spa
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['teacher_prefix'] = cat_list


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

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

1.1.5 Preprocessing of project_grade_category

In [10]:

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

project_data['project_grade_category'] = cat_list

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

# Citation code : https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-sciki
# To convert the data type object to unicode string : used """astype('U')""" code from 
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039

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



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

1.2. Text Preprocessing

1.2.1 Text Preprocessing of essay

In [11]:

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

In [12]:

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

Out[12]:

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

In [13]:

```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*125)
print(project_data['essay'].values[150])
print("="*125)
print(project_data['essay'].values[2019])
print("="*125)
print(project_data['essay'].values[30019])
print("="*125)
print(project_data['essay'].values[99999])
print("="*125)
```

```
I have been fortunate enough to use the Fairy Tale STEM kits in my classro
om as well as the STEM journals, which my students really enjoyed.  I woul
d love to implement more of the Lakeshore STEM kits in my classroom for th
e next school year as they provide excellent and engaging STEM lessons.My
students come from a variety of backgrounds, including language and socioe
conomic status.  Many of them don't have a lot of experience in science an
d engineering and these kits give me the materials to provide these exciti
ng opportunities for my students.Each month I try to do several science or
STEM/STEAM projects.  I would use the kits and robot to help guide my scie
nce instruction in engaging and meaningful ways.  I can adapt the kits to
my current language arts pacing guide where we already teach some of the m
aterial in the kits like tall tales (Paul Bunyan) or Johnny Appleseed.  Th
e following units will be taught in the next school year where I will impl
ement these kits: magnets, motion, sink vs. float, robots.  I often get to
these units and don't know If I am teaching the right way or using the rig
ht materials.    The kits will give me additional ideas, strategies, and l
essons to prepare my students in science.It is challenging to develop high
quality science activities.  These kits give me the materials I need to pr
ovide my students with science activities that will go along with the curr
iculum in my classroom.  Although I have some things (like magnets) in my
classroom, I don't know how to use them effectively.  The kits will provid
e me with the right amount of materials and show me how to use them in an
appropriate way.
==========================================================================
===================================================
I teach high school English to students with learning and behavioral disab
ilities. My students all vary in their ability level. However, the ultimat
e goal is to increase all students literacy levels. This includes their re
ading, writing, and communication levels.I teach a really dynamic group of
students. However, my students face a lot of challenges. My students all l
ive in poverty and in a dangerous neighborhood. Despite these challenges,
I have students who have the the desire to defeat these challenges. My stu
dents all have learning disabilities and currently all are performing belo
w grade level. My students are visual learners and will benefit from a cla
ssroom that fulfills their preferred learning style.The materials I am req
uesting will allow my students to be prepared for the classroom with the n
ecessary supplies.  Too often I am challenged with students who come to sc
hool unprepared for class due to economic challenges.  I want my students
to be able to focus on learning and not how they will be able to get schoo
l supplies.  The supplies will last all year.  Students will be able to co
mplete written assignments and maintain a classroom journal.  The chart pa
per will be used to make learning more visual in class and to create poste
rs to aid students in their learning.  The students have access to a class
room printer.  The toner will be used to print student work that is comple
ted on the classroom Chromebooks.I want to try and remove all barriers for
the students learning and create opportunities for learning. One of the bi
ggest barriers is the students not having the resources to get pens, pape
```

```
r, and folders. My students will be able to increase their literacy skills
because of this project.
==========================================================================
===================================================
We are 32 super second graders who are trying to use technology as often a
s possible!Many of my students come from a low income home with parents wh
o have limited, if any, English skills.\r\n\r\nMany students are still lea
rning English themselves! The students do not receive the help they need a
t home because parents don't understand the skills or they're busy workin
g. Both the students and parents WANT to succeed, they just need additiona
l help. They are eager and excited to learn and be productive members of o
ur school and community. At our school, and within the classroom we strive
to not only teach our students reading, writing, math, etc...but to be goo
d people with kind hearts that contribute to their community.We have Chrom
ebooks in our class and the earbuds would greatly help in 2 ways. \r\nOur
school has access to a program that will allow kids to not only listen to
books, but to read them back to me via the program.  This is a great oppor
tunity for me to listen to each student and modify their learning. Time do
esn't permit me to listen to each student read an entire book each day.\r
\nAdditionally, I'd like to use the students to do research, but much of w
hat we are researching requires the kids to listen to the information.  It
is quite frustrating for kids right now because when we have multiple Chro
mebooks in use, the classroom gets pretty loud and distracting.Although mo
st of my students can read fairly well, having books read to them is a gre
at modeling opportunity that they don't have at home.  In addition, they c
an then read the book back to me.\r\nThey are also learning how to do rese
arch and gather information online.  This is hard task with low or no volu
me.  Earbuds would allow them to do research while letting me work with ot
her groups distraction free!
==========================================================================
===================================================
I am so grateful to begin another exciting school year at Stafford Primar
y.  Our theme is Superhero, and we are having fun with this theme preparin
g or classrooms for our little Superheroes this school year. Our school is
very unique in the our diverse staff and student body. Our school district
Stafford Municipal School District can best be described as one happy fami
ly.  33 languages are spoken throughout the district.  My students are spe
cial because they are immersed in an environment where we teach them their
academic and how to embrace the diverse student body.   Learning is happen
ing all over our school. In the classroom students are learning required c
ontent however, outside the classroom, students have art, physical educati
on, and music.  The students have opportunities come to see the ballet, vi
sit the animals at the zoo, heritage assemblies, etc.  I always say there
is never a dull moment at Stafford Primary School and the students enjoy t
heir time at school.This year of school district has decided to use Math T
alks to help students to understand how to apply math concepts. Math Talks
includes hands on and mental math concepts.  Many of my students struggle
with basic math concepts and my goal as their teacher is to help them to u
nderstand the math we are teaching why they are learning, most important h
ow to apply math to their personal experiences.  I feared math in school a
nd I want to take that fear away from my students by making math fun and e
ngaging for them.  Our reading initiative Literacy for All will help all s
tudents excel in literacy. As a kindergarten teacher it is my obligation t
o set a strong foundation so their journey in school is exciting and succe
ssful. \r\n\r\nThe iPad covers and headphones are needed to protect the iP
ads I recived from my debut project.  The covers will protect the iPads fr
om damage, and the headphones will help keep the noise level down in the c
lassroom and cut down on distracting other students. \r\nAs stated in my i
ntroduction we are implementing Number Talks and Literacy for All . The ma
terials I am requesting will help students connect and learning strategies
I will teach them so they can apply their knowledge independently during t
```

heir workstation time . During workstation time students work independentl
y while I am in small group assisting students who are struggling as well
as those who are advanced and are ready to move on in their learning.Mrs.M
rs.
===
===
My classroom consists of twenty-two amazing sixth graders from different c
ultures and backgrounds. They are a social bunch who enjoy working in part
ners and working with groups. They are hard-working and eager to head to m
iddle school next year. My job is to get them ready to make this transitio
n and make it as smooth as possible. In order to do this, my students need
to come to school every day and feel safe and ready to learn. Because they
are getting ready to head to middle school, I give them lots of choice- ch
oice on where to sit and work, the order to complete assignments, choice o
f projects, etc. Part of the students feeling safe is the ability for them
to come into a welcoming, encouraging environment. My room is colorful and
the atmosphere is casual. I want them to take ownership of the classroom b
ecause we ALL share it together. Because my time with them is limited, I w
ant to ensure they get the most of this time and enjoy it to the best of t
heir abilities.Currently, we have twenty-two desks of differing sizes, yet
the desks are similar to the ones the students will use in middle school.
We also have a kidney table with crates for seating. I allow my students t
o choose their own spots while they are working independently or in group
s. More often than not, most of them move out of their desks and onto the
crates. Believe it or not, this has proven to be more successful than maki
ng them stay at their desks! It is because of this that I am looking towar
d the “Flexible Seating” option for my classroom.\r\n The students look fo
rward to their work time so they can move around the room. I would like to
get rid of the constricting desks and move toward more “fun” seating optio
ns. I am requesting various seating so my students have more options to si
t. Currently, I have a stool and a papasan chair I inherited from the prev
ious sixth-grade teacher as well as five milk crate seats I made, but I wo
uld like to give them more options and reduce the competition for the “goo
d seats”. I am also requesting two rugs as not only more seating options b
ut to make the classroom more welcoming and appealing. In order for my stu
dents to be able to write and complete work without desks, I am requesting
a class set of clipboards. Finally, due to curriculum that requires groups
to work together, I am requesting tables that we can fold up when we are n
ot using them to leave more room for our flexible seating options.\r\nI kn
ow that with more seating options, they will be that much more excited abo
ut coming to school! Thank you for your support in making my classroom one
students will remember forever!Mrs.Mrs.
===
===

In [14]:

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

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

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

In [15]:

```
sent = decontracted(project_data['essay'].values[2019])
print(sent)
print("="*125)
```

```
We are 32 super second graders who are trying to use technology as often a
s possible!Many of my students come from a low income home with parents wh
o have limited, if any, English skills.\r\n\r\nMany students are still lea
rning English themselves! The students do not receive the help they need a
t home because parents do not understand the skills or they are busy worki
ng. Both the students and parents WANT to succeed, they just need addition
al help. They are eager and excited to learn and be productive members of
our school and community. At our school, and within the classroom we striv
e to not only teach our students reading, writing, math, etc...but to be g
ood people with kind hearts that contribute to their community.We have Chr
omebooks in our class and the earbuds would greatly help in 2 ways. \r\nOu
r school has access to a program that will allow kids to not only listen t
o books, but to read them back to me via the program.  This is a great opp
ortunity for me to listen to each student and modify their learning. Time
does not permit me to listen to each student read an entire book each da
y.\r\nAdditionally, I would like to use the students to do research, but m
uch of what we are researching requires the kids to listen to the informat
ion.  It is quite frustrating for kids right now because when we have mult
iple Chromebooks in use, the classroom gets pretty loud and distracting.Al
though most of my students can read fairly well, having books read to them
is a great modeling opportunity that they do not have at home.  In additio
n, they can then read the book back to me.\r\nThey are also learning how t
o do research and gather information online.  This is hard task with low o
r no volume.  Earbuds would allow them to do research while letting me wor
k with other groups distraction free!
==========================================================================
===================================================
```

In [16]:

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

```
We are 32 super second graders who are trying to use technology as often a
s possible Many of my students come from a low income home with parents wh
o have limited, if any, English skills.    Many students are still learnin
g English themselves  The students do not receive the help they need at ho
me because parents do not understand the skills or they are busy working.
Both the students and parents WANT to succeed, they just need additional h
elp. They are eager and excited to learn and be productive members of our
school and community. At our school, and within the classroom we strive to
not only teach our students reading, writing, math, etc...but to be good p
eople with kind hearts that contribute to their community.We have Chromebo
oks in our class and the earbuds would greatly help in 2 ways.   Our schoo
l has access to a program that will allow kids to not only listen to book
s, but to read them back to me via the program.  This is a great opportuni
ty for me to listen to each student and modify their learning. Time does n
ot permit me to listen to each student read an entire book each day.  Addi
tionally, I would like to use the students to do research, but much of wha
t we are researching requires the kids to listen to the information.  It i
s quite frustrating for kids right now because when we have multiple Chrom
ebooks in use, the classroom gets pretty loud and distracting.Although mos
t of my students can read fairly well, having books read to them is a grea
t modeling opportunity that they do not have at home.  In addition, they c
an then read the book back to me.  They are also learning how to do resear
ch and gather information online.  This is hard task with low or no volum
e.  Earbuds would allow them to do research while letting me work with oth
er groups distraction free
```

In [17]:

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

```
We are 32 super second graders who are trying to use technology as often a
s possible Many of my students come from a low income home with parents wh
o have limited if any English skills Many students are still learning Engl
ish themselves The students do not receive the help they need at home beca
use parents do not understand the skills or they are busy working Both the
students and parents WANT to succeed they just need additional help They a
re eager and excited to learn and be productive members of our school and
community At our school and within the classroom we strive to not only tea
ch our students reading writing math etc but to be good people with kind h
earts that contribute to their community We have Chromebooks in our class
and the earbuds would greatly help in 2 ways Our school has access to a pr
ogram that will allow kids to not only listen to books but to read them ba
ck to me via the program This is a great opportunity for me to listen to e
ach student and modify their learning Time does not permit me to listen to
each student read an entire book each day Additionally I would like to use
the students to do research but much of what we are researching requires t
he kids to listen to the information It is quite frustrating for kids righ
t now because when we have multiple Chromebooks in use the classroom gets
pretty loud and distracting Although most of my students can read fairly w
ell having books read to them is a great modeling opportunity that they do
not have at home In addition they can then read the book back to me They a
re also learning how to do research and gather information online This is
hard task with low or no volume Earbuds would allow them to do research wh
ile letting me work with other groups distraction free
```

In [18]:

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

In [19]:

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

```
  0%|          | 0/109248 [00:00<?, ?it/s]

CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 6.91 µs

100%|██████████| 109248/109248 [03:22<00:00, 539.26it/s]
```

In [20]:

```
# after preprocesing
preprocessed_essays[30019]
```

Out[20]:

```
'grateful exciting school year stafford primary theme superhero fun theme
preparing classrooms superheroes school year school unique diverse staff s
tudent body school district stafford municipal school district happy famil
y 33 languages spoken district students special immersed environment teach
academic embrace diverse student body learning happening school classroom
students learning required content classroom students art physical educati
on music students opportunities ballet visit animals zoo heritage assembli
es dull moment stafford primary school students enjoy time school year sch
ool district decided math talks students understand apply math concepts ma
th talks includes hands mental math concepts students struggle basic math
concepts goal teacher understand math teaching learning apply math persona
l experiences feared math school fear students making math fun engaging re
ading initiative literacy students excel literacy kindergarten teacher obl
igation set strong foundation journey school exciting successful ipad cove
rs headphones needed protect ipads recived debut project covers protect ip
ads damage headphones noise level classroom cut distracting students state
d introduction implementing number talks literacy materials requesting stu
dents connect learning strategies teach apply knowledge independently work
station time workstation time students work independently small group assi
sting students struggling advanced ready move learning'
```

1.2.2 Text Preprocessing of project_title

In [21]:

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

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

In [22]:

```
# printing some random title texts
print(project_data['project_title'].values[19])
print('--'*19)
print(project_data['project_title'].values[198])
print('--'*19)
print(project_data['project_title'].values[1989])
print('--'*19)
print(project_data['project_title'].values[99999])
print('--'*19)
```

```
Choice Novels for Freshman Students are Needed!!!!!!
--------------------------------------
Expressions of Learning
--------------------------------------
Kindergarten Wants to Be Zookeepers with Letters Alive!
--------------------------------------
Turning to Flexible Seating: One Sixth-Grade Class's Journey to Freedom
--------------------------------------
```

In [23]:

```
# 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 [24]:

```
sent = decontracted(project_data['project_title'].values[99999])
print(sent)
print("="*125)
```

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

In [25]:

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

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

In [26]:

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

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

In [27]:

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

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

In [28]:

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

```
  0%|          | 0/109248 [00:00<?, ?it/s]

CPU times: user 0 ns, sys: 0 ns, total: 0 ns
Wall time: 6.44 µs

100%|██████████| 109248/109248 [00:07<00:00, 15578.76it/s]
```

In [29]:

```
preprocessed_project_title[99991]
```

Out[29]:

```
'media literacy project students severe profound disabilities'
```

1.3. Numerical normalization

1.3.1 normalization_price

In [30]:

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

Out[30]:

```
(109248, 20)
```

In [31]:

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

Out[31]:

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

In [32]:

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

```
(109248,)
```

In [33]:

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

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

1.3.2 Normalization of teacher_number_of_previously_posted_projects

In [34]:

```
project_data["teacher_number_of_previously_posted_projects"].values
```

Out[34]:

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

In [35]:

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

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

1.3.3 spilt the data into train ,CV and test

In [36]:

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

Out[36]:

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

In [37]:

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

Out[37]:

```
array([1, 1, 1, ..., 1, 1, 1])
```

In [38]:

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

In [39]:

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

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

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

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

1.4. Vectorizing Categorical data

1.4.1 Vectorization of project_subject_categories¶

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

In [40]:

```
# 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, 

clean_categories_one_hot_train = vectorizer.fit_transform(X_train['clean_categories'].v
clean_categories_one_hot_test  = vectorizer.transform(X_test['clean_categories'].values
clean_categories_one_hot_cv    = vectorizer.transform(X_cv['clean_categories'].values)

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

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

1.4.2 Vectorization of project_subject_subcategories

In [41]:

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

clean_subcategories_one_hot_train = vectorizer.fit_transform(X_train['clean_subcategori
clean_subcategories_one_hot_test  = vectorizer.transform(X_test['clean_subcategories'].
clean_subcategories_one_hot_cv    = vectorizer.transform(X_cv['clean_subcategories'].va

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

```
vectorizer feature names : ['College_CareerPrep', 'ESL', 'TeamSports', 'Ca
re_Hunger', 'SpecialNeeds', 'PerformingArts', 'SocialSciences', 'Music',
'Literature_Writing', 'Other', 'EnvironmentalScience', 'EarlyDevelopment',
'Extracurricular', 'Economics', 'VisualArts', 'Gym_Fitness', 'FinancialLit
eracy', 'History_Geography', 'Health_LifeScience', 'Health_Wellness', 'Mat
hematics', 'Literacy', 'ForeignLanguages', 'Warmth', 'CommunityService',
'Civics_Government', 'AppliedSciences', 'ParentInvolvement', 'CharacterEdu
cation', 'NutritionEducation']
--------------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 30)
Shape of matrix after one hot encodig test  :  (36052, 30)
Shape of matrix after one hot encodig cv    :  (24155, 30)
```

1.4.3 Vectorization of school_state

In [42]:

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

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

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

```
['FL', 'OR', 'NV', 'LA', 'MI', 'OK', 'HI', 'AL', 'VA', 'ID', 'IA', 'AZ',
'VT', 'OH', 'WA', 'ND', 'MA', 'AK', 'TX', 'NE', 'PA', 'NY', 'IN', 'CO', 'M
O', 'WY', 'KS', 'NJ', 'DE', 'CA', 'NC', 'CT', 'UT', 'MD', 'KY', 'AR', 'D
C', 'RI', 'MN', 'IL', 'NH', 'NM', 'GA', 'SD', 'MT', 'WI', 'SC', 'WV', 'M
E', 'MS', 'TN']
--------------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 51)
Shape of matrix after one hot encodig test  :  (36052, 51)
Shape of matrix after one hot encodig cv    :  (24155, 51)
```

1.4.4 Vectorization of teacher_prefix

In [43]:

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


teacher_prefix_one_hot_train = vectorizer.fit_transform(X_train['teacher_prefix'].value
teacher_prefix_one_hot_test  = vectorizer.transform(X_test['teacher_prefix'].values.ast
teacher_prefix_one_hot_cv    = vectorizer.transform(X_cv['teacher_prefix'].values.astyp

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

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

In [44]:

```
vectorizer.get_feature_names
```

Out[44]:

```
<bound method CountVectorizer.get_feature_names of CountVectorizer(analyze
r='word', binary=True, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='cont
ent',
                lowercase=False, max_df=1.0, max_features=None, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None,
                vocabulary=['Ms.', 'Teacher', 'Mrs.', 'Dr.', 'Mr.'])>
```

1.4.5 Vectorization of project_grade_category

In [45]:

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

project_grade_category_one_hot_train = vectorizer.fit_transform(X_train['project_grade_
project_grade_category_one_hot_test  = vectorizer.transform(X_test['project_grade_categ
project_grade_category_one_hot_cv    = vectorizer.transform(X_cv['project_grade_categor

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

```
['Grades9-12', 'Grades3-5', 'GradesPreK-2', 'Grades6-8']
---------------------------------------------------------
Shape of matrix after one hot encodig train  :  (49041, 4)
Shape of matrix after one hot encodig test   :  (36052, 4)
Shape of matrix after one hot encodig cv     :  (24155, 4)
```

1.5. Vectorizing Text

1.5.1 Vectorization of essays bow

In [46]:

```
X_train['essay'].tail(2)
```

Out[46]:

```
92981    We are a Title I school with 90% of our studen...
41050    Children have a hard time focusing and learnin...
Name: essay, dtype: object
```

In [47]:

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

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

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

```
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 12568)
Shape of matrix after one hot encodig test  :  (36052, 12568)
Shape of matrix after one hot encodig cv    :  (24155, 12568)
```

In [48]:

```
vectorizer_essays_bow.get_feature_names
```

Out[48]:

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

1.5.1.1 Vectorization of essays tfidf

In [49]:

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

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

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

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

1.5.1.2 Using Pretrained Models: essays Avg W2V

In [50]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[50]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/3823034
9/4084039\ndef (https://stackoverflow.com/a/38230349/4084039\ndef) loadGlo
veModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(glove
File,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n
splitLine = line.split()\n        word = splitLine[0]\n        embedding =
np.array([float(val) for val in splitLine[1:]])\n        model[word] = emb
edding\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\nword
s = []\nfor i in preproced_essays:\n    words.extend(i.split(\' \'))\n\nfo
r i in preprocessed_project_title:\n    words.extend(i.split(\' \'))\nprin
t("all the words in the coupus", len(words))\nwords = set(words)\nprint("t
he unique words in the coupus", len(words))\n\ninter_words = set(model.key
s()).intersection(words)\nprint("The number of words that are present in b
oth glove vectors and our coupus",       len(inter_words),"(",np.round(len
(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove =
set(model.keys())\nfor i in words:\n    if i in words_glove:\n        word
s_courpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n
\n\n# stronging variables into pickle files python: http://www.jessicayun
g.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport (h
ttp://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-
python/\n\nimport) pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n
pickle.dump(words_courpus, f)\n\n\n'
```

In [51]:

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

In [52]:

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

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

```
100%|██████████| 49041/49041 [00:23<00:00, 2097.53it/s]

49041
300
```

Average Word2Vec X_test_essay

In [53]:

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

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

```
100%|██████████| 36052/36052 [00:17<00:00, 2103.03it/s]

36052
300
```

Average Word2Vec X_cv_essay

In [54]:

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

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

```
100%|██████████| 24155/24155 [00:11<00:00, 2122.63it/s]

24155
300
```

1.5.1.3 essays TFIDF weighted W2V train

In [55]:

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

In [56]:

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

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

```
100%|██████████| 49041/49041 [03:23<00:00, 241.34it/s]

49041
300
```

essays TFIDF weighted W2V test

In [57]:

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

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

```
100%|██████████| 36052/36052 [02:21<00:00, 245.33it/s]

36052
300
```

essays TFIDF weighted W2V cv

In [58]:

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

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

```
100%|██████████| 24155/24155 [01:32<00:00, 261.52it/s]

24155
300
```

1.5.2 Vectorization of project_title bow train, test, cv

In [59]:

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

Out[59]:

```
32219    How Can I Learn to Use the Computer?
Name: project_title, dtype: object
```

In [60]:

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

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

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

```
---------------------------------------------------------
Shape of matrix after one hot encodig train :  (49041, 2124)
Shape of matrix after one hot encodig test  :  (36052, 2124)
Shape of matrix after one hot encodig cv    :  (24155, 2124)
```

1.5.2.1 Vectorization of project_title tfidf train, test. cv

In [61]:

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

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

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

```
Shape of matrix after one hot encodig of train :  (49041, 2124)
Shape of matrix after one hot encodig test     :  (36052, 2124)
Shape of matrix after one hot encodig cv       :  (24155, 2124)
```

1.5.2.2 Using Pretrained Models: project_title Avg W2V train

In [62]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[62]:

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

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

In [63]:

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

In [64]:

```
# average Word2Vec  project_title_train
# compute average word2vec for each review.
project_title_avg_w2v_vectors_train = []; # the avg-w2v for each sentence/review is sto
for sentence in tqdm(X_train['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    project_title_avg_w2v_vectors_train.append(vector)

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

```
100%|██████████| 49041/49041 [00:00<00:00, 108329.93it/s]

49041
300
```

Average Word2Vec project_title_test

In [65]:

```
# average Word2Vec  project_title_test
# compute average word2vec for each review.
project_title_avg_w2v_vectors_test = []; # the avg-w2v for each sentence/review is stor
for sentence in tqdm(X_test['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    project_title_avg_w2v_vectors_test.append(vector)

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

```
100%|██████████| 36052/36052 [00:00<00:00, 104941.58it/s]

36052
300
```

Average Word2Vec project_title_cv

In [66]:

```
# average Word2Vec  project_title_ cv
# compute average word2vec for each review.
project_title_avg_w2v_vectors_cv = []; # the avg-w2v for each sentence/review is stored
for sentence in tqdm(X_cv['project_title']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    project_title_avg_w2v_vectors_cv.append(vector)

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

```
100%|██████████| 24155/24155 [00:00<00:00, 102331.61it/s]

24155
300
```

1.5.2.3 project_title TFIDF weighted W2V train

In [67]:

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

In [68]:

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

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

```
100%|██████████| 49041/49041 [00:00<00:00, 73742.72it/s]

49041
300
```

project_title TFIDF weighted W2V_ test

In [69]:

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

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

```
100%|██████████| 36052/36052 [00:00<00:00, 71664.78it/s]

36052
300
```

project_title TFIDF weighted W2V_cv

In [70]:

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

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

```
100%|██████████| 24155/24155 [00:00<00:00, 65319.92it/s]

24155
300
```

1.6. Vectorizing Numerical features

1.6.1 Normalization of price_train_test_cv

In [71]:

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

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


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

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

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

```
shape of price_normalizer_train: (49041, 1)
---------------------------
[[7.69688242e-03]
 [1.52298259e-03]
 [8.71107242e-05]
 ...
 [1.59961675e-03]
 [1.51570720e-03]
 [2.51709251e-03]]
shape of price_normalizer_test : (36052, 1)
---------------------------
[[0.00144681]
 [0.00110718]
 [0.00984206]
 ...
 [0.00011076]
 [0.00344277]
 [0.0021036 ]]
shape of price_normalizer_cv   : (24155, 1)
---------------------------
[[0.00345492]
 [0.00221106]
 [0.00164549]
 ...
 [0.00140978]
 [0.00218327]
 [0.00260149]]
```

1.6.2 Teacher_number_of_previously_posted_projects train_test_cv :

[illegible]
Numerical / Normalization

In [72]:

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

teacher_number_of_previously_posted_projects_normalizer_train = teacher_number_of_previ

teacher_number_of_previously_posted_projects_normalizer_test  = teacher_number_of_previ

teacher_number_of_previously_posted_projects_normalizer_cv    = teacher_number_of_previ


teacher_number_of_previously_posted_projects_normalizer_train=teacher_number_of_previou
teacher_number_of_previously_posted_projects_normalizer_test=teacher_number_of_previous
teacher_number_of_previously_posted_projects_normalizer_cv = teacher_number_of_previous



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

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

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

```
shape of teacher_number_of_previously_posted_projects_normalizer_train:
(49041, 1)
---------------------------
[[0.00045253]
 [0.        ]
 [0.        ]
 ...
 [0.00256436]
 [0.00226267]
 [0.00497787]]
shape of teacher_number_of_previously_posted_projects_normalizer_test :
(36052, 1)
---------------------------
[[0.00210234]
 [0.00140156]
 [0.        ]
 ...
 [0.05816465]
 [0.00140156]
 [0.00140156]]
shape of teacher_number_of_previously_posted_projects_normalizer_cv   :
(24155, 1)
---------------------------
[[0.00021641]
 [0.00216408]
 [0.00627583]
 ...
 [0.00086563]
 [0.00216408]
 [0.00259689]]
```

In [ ]:

```

```

In [73]:

```
project_data.columns
```

Out[73]:

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

1.7 Computing Sentiment Scores essay

In [74]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# Link :https://stackoverflow.com/questions/53680690/list-object-has-no-attribute-encod

vader = SentimentIntensityAnalyzer()
essay_train_str = X_train['essay'].str[0].str.join(" ")
sse = essay_train_str.apply(lambda x: vader.polarity_scores(x)['compound']) # sse = Sen


# Normalization

normalizer = Normalizer()
sse_train = normalizer.fit_transform(sse.values.reshape(-1,1))
print(sse_train)

print("Shape of sse_train and y_train :", sse_train.shape, y_train.shape)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
Shape of sse_train and y_train : (49041, 1) (49041,)
```

In [75]:

```
# Sentiment Scores essay : test sse_test

vader = SentimentIntensityAnalyzer()
essay_test_str = X_test['essay'].str[0].str.join(" ")
sse = essay_test_str.apply(lambda x: vader.polarity_scores(x)['compound']) # sse = Sent


# Normalization

normalizer = Normalizer()
sse_test = normalizer.fit_transform(sse.values.reshape(-1,1))
print(sse_test)

print("Shape of sse_test and y_test :", sse_test.shape, y_test.shape)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
Shape of sse_test and y_test : (36052, 1) (36052,)
```

In [76]:

```
# Sentiment Scores essay : test sse_cv

vader = SentimentIntensityAnalyzer()
essay_cv_str = X_cv['essay'].str[0].str.join(" ")
sse = essay_cv_str.apply(lambda x: vader.polarity_scores(x)['compound']) # sse = Sentim


# Normalization

normalizer = Normalizer()
sse_cv = normalizer.fit_transform(sse.values.reshape(-1,1))
print(sse_cv)

print("Shape of sse_cv and y_cv :", sse_cv.shape, y_cv.shape)
```

```
[[0.]
 [0.]
 [0.]
 ...
 [0.]
 [0.]
 [0.]]
Shape of sse_cv and y_cv : (24155, 1) (24155,)
```

Assignment : SVM

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

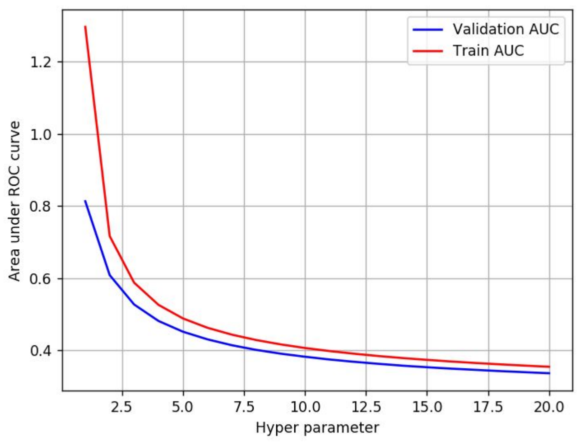
    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)
    - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: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

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

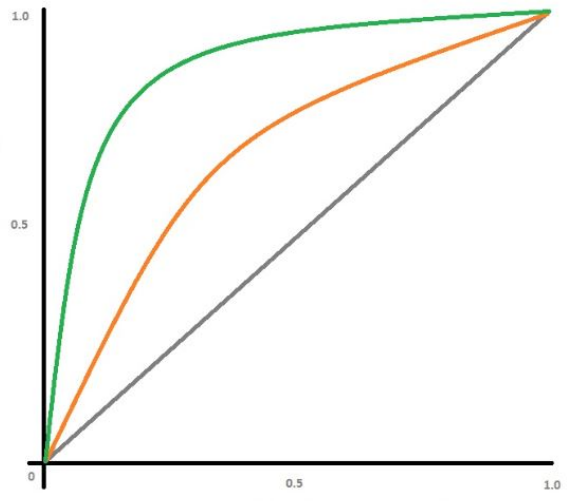
    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data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

3. **Representation of results**

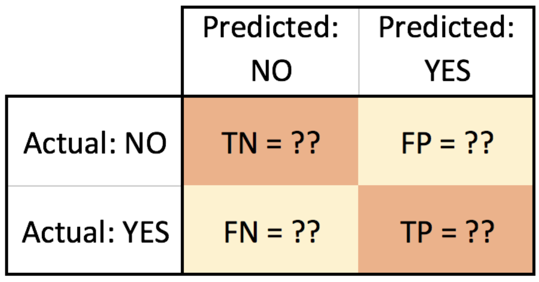
    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure.



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



- Along with plotting ROC curve, you need to print the confusion matrix (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.



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

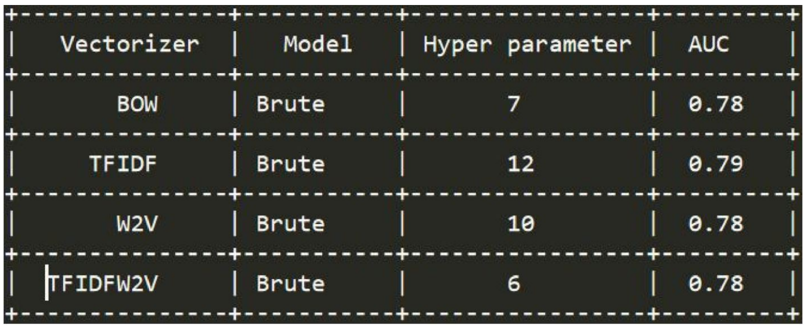
(https://seaborn.pydata.org/generated/seaborn.heatmap.html)

4. **[Task-2] Apply the Support Vector Machines on these features by finding the best hyper paramter as suggested in step 2 and step 3**

    - Consider these set of features Set 5 :
        - **school_state** : categorical data
        - **clean_categories** : categorical data
        - **clean_subcategories** : categorical data
        - **project_grade_category** :categorical data
        - **teacher_prefix** : categorical data
        - **quantity** : numerical data
        - **teacher_number_of_previously_posted_projects** : numerical data
        - **price** : numerical data

- **sentiment score's of each of the essay** : numerical data
- **number of words in the title** : numerical data
- **number of words in the combine essays** : numerical data
- **Apply [TruncatedSVD (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html)](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)](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/)](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/)



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

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 Applying Support Vector Machines on BOW, SET 1

Merging features encoding numerical + categorical features BOW, SET 1

In [77]:

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

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


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

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

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

```
Final Data Matrix of set1 :
shape of set1_train and y_train : (49041, 14793) (49041,)
shape of set1_test and y_test   : (36052, 14793) (36052,)
shape of set1_cv and y_cv       : (24155, 14793) (24155,)
```

2.1.1.A Hyper parameter Tuning to find alpha:: Simple cross Validation set1

In [78]:

```
%%time
from sklearn import linear_model
from sklearn.metrics import accuracy_score
params = [10**-4, 10**-2, 1, 10, 10**2,  10**4]
for i in params:
    # instantiate learning model
    clf = linear_model.SGDClassifier(loss='hinge', penalty='l2', alpha = i, class_weigh
    # fitting the model on crossvalidation train

    clf.fit(set1_train, y_train)
    # predict the response on the crossvalidation train

    pred = clf.predict(set1_cv)
    # evaluate CV accuracy

    acc = accuracy_score(y_cv, pred, normalize=True) * float(100)
    print('\nCV accuracy for alpha = %d is %d%%' % (i, acc))
```

```
CV accuracy for alpha = 0 is 67%

CV accuracy for alpha = 0 is 70%

CV accuracy for alpha = 1 is 69%

CV accuracy for alpha = 10 is 84%

CV accuracy for alpha = 100 is 84%

CV accuracy for alpha = 10000 is 84%
CPU times: user 16.9 s, sys: 2.45 s, total: 19.3 s
Wall time: 14.5 s
```

In [79]:

```
clf.get_params
```

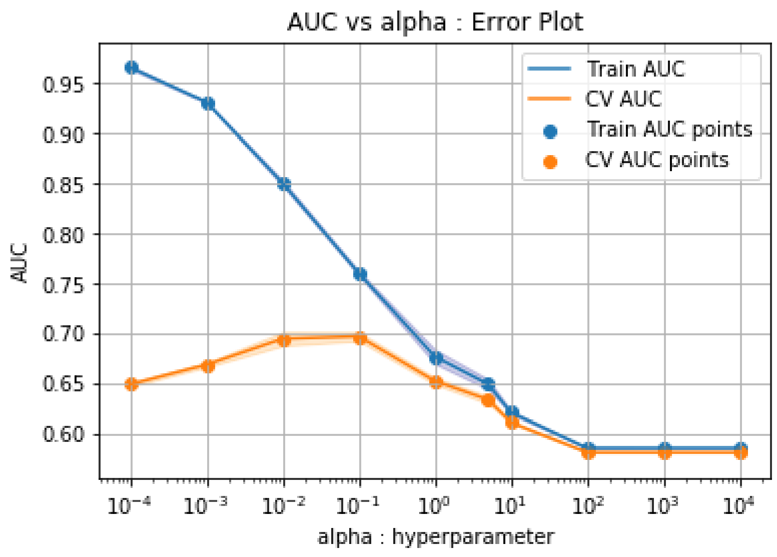
Out[79]:

```
<bound method BaseEstimator.get_params of SGDClassifier(alpha=10000, avera
ge=False, class_weight='balanced',
              early_stopping=False, epsilon=0.1, eta0=0.0, fit_intercept=T
rue,
              l1_ratio=0.15, learning_rate='optimal', loss='hinge',
              max_iter=1000, n_iter_no_change=5, n_jobs=None, penalty='l
2',
              power_t=0.5, random_state=None, shuffle=True, tol=0.001,
              validation_fraction=0.1, verbose=0, warm_start=False)>
```

2.1.1 B. Hyper parameter Tuning to find alpha :: RandomizedSearchCV set_1

In [80]:

```
%%time
# Link : https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.Rand

from sklearn.model_selection import RandomizedSearchCV
from sklearn import linear_model
svm = linear_model.SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
alpha = [0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 10000]
params = {'alpha':[0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 10000]}
clf_1 = RandomizedSearchCV(svm, params,cv=2,scoring='roc_auc', return_train_score=True)
clf_1.fit(set1_train, y_train)

train_auc     = clf_1.cv_results_['mean_train_score']
train_auc_std = clf_1.cv_results_['std_train_score']
cv_auc        = clf_1.cv_results_['mean_test_score']
cv_auc_std    = clf_1.cv_results_['std_test_score']

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

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

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

plt.legend()
plt.grid(True)
plt.xscale('log')
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs alpha : Error Plot")
plt.show()
```



```
CPU times: user 34.3 s, sys: 10.3 s, total: 44.7 s
Wall time: 27 s
```

In [81]:

```
clf_1.best_params_
```

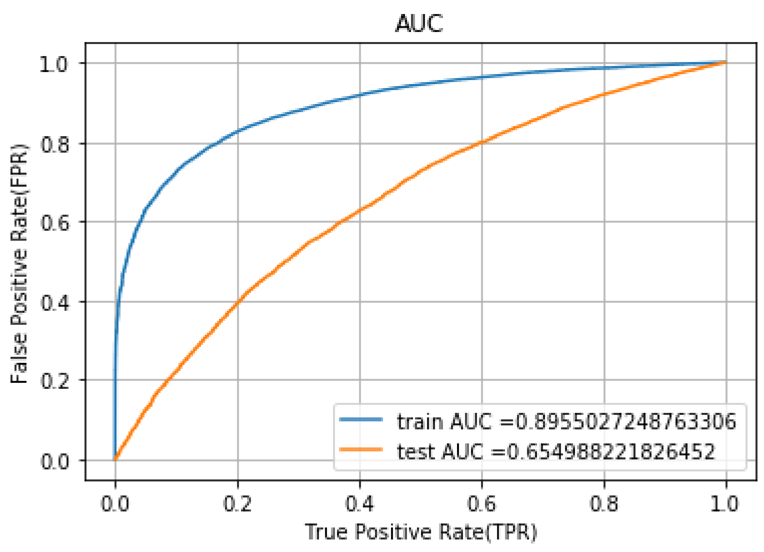
Out[81]:

```
{'alpha': 0.1}
```

2.1.2 Train model using the best hyper-parameter(alpha) value set1

In [85]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
# link : https://github.com/scikit-learn/scikit-learn/issues/7278
from sklearn import linear_model
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

clf = linear_model.SGDClassifier(alpha = 0.1,loss = 'hinge', class_weight='balanced')
svm = CalibratedClassifierCV(cv=2, method='sigmoid').fit(set1_train, y_train)


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

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

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

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



```
CPU times: user 48 s, sys: 1.33 s, total: 49.3 s
Wall time: 46.8 s
```

2.1.3. Confustion Matrix set1_train and set1_test

In [86]:

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

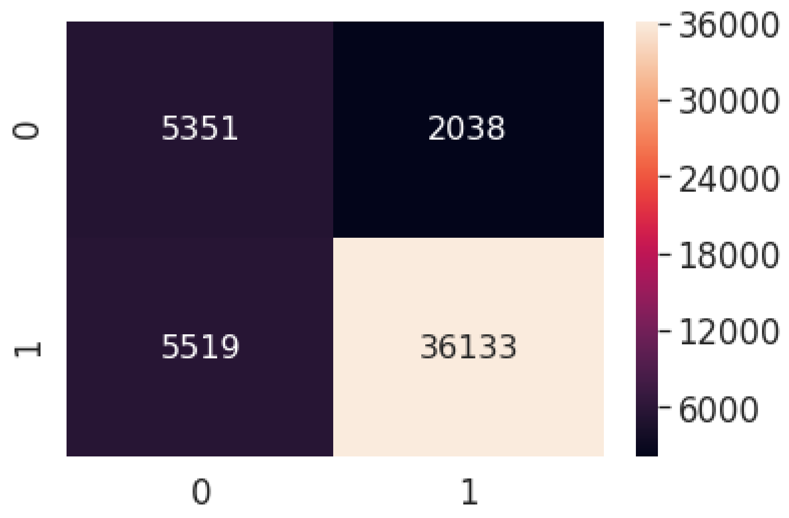
In [87]:

```
#Confustion Matrix Set1_train

svm.fit(set1_train,y_train)
y_train_pred_1 = svm.predict_proba(set1_train)[:,1]
conf_matr_df_train_1 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_1,thr


sns.set(font_scale=1.5)#for label size
sns.heatmap(conf_matr_df_train_1, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999542102075 for threshold 0.822
```

Out[87]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa05efac8>
```



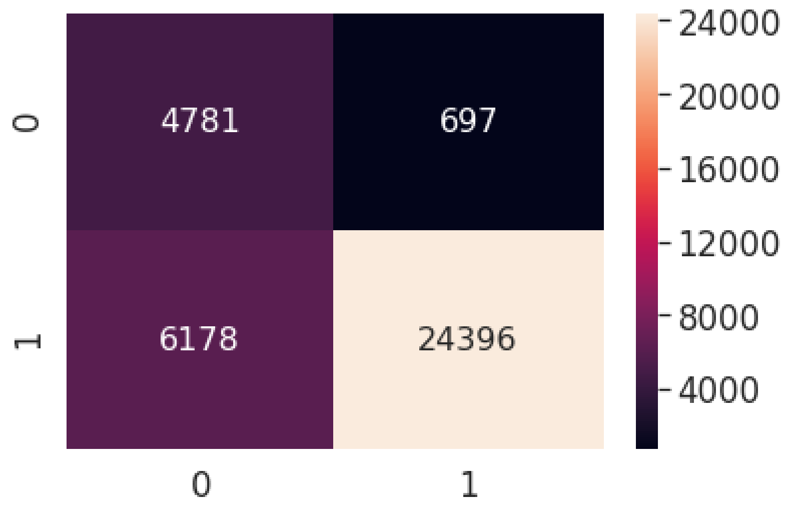
In [88]:

```
# Confusion Matrix set1_test
svm.fit(set1_test, y_test)
y_test_pred_1 = svm.predict_proba(set1_test)[:, 1]
conf_matr_df_test_1 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred_1, thre

sns.set(font_scale=1.5) # Label Size
sns.heatmap(conf_matr_df_test_1, annot=True, annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.831
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa60b5dd8>
```



2.2 Applying SVM on TFIDF, SET 2

Merging features encoding numerical + categorical features TFIDF, SET 2

In [89]:

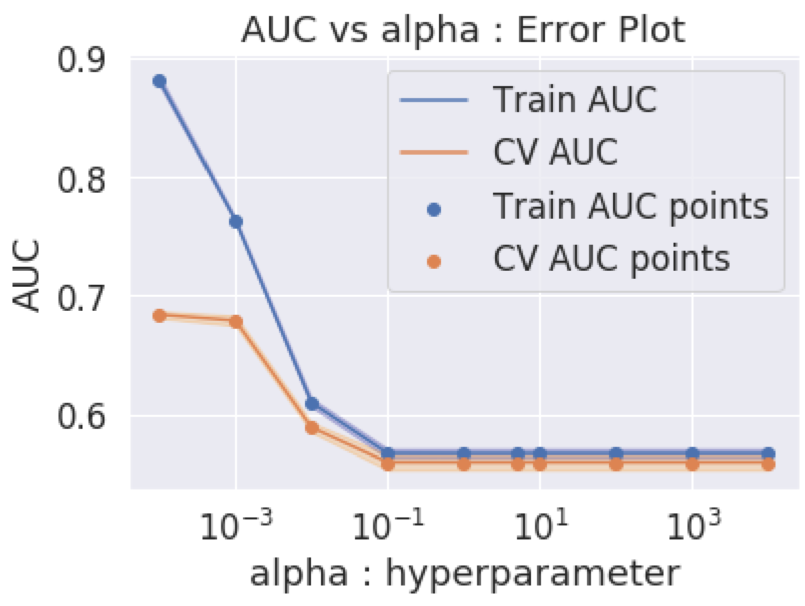
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set2_ = all categorical features + numarical features + essays_tfidf + project_title_

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


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

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

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

```
Final Data Matrix of set2 :
shape of set2_train and y_train : (49041, 7225) (49041,)
shape of set2_test and y_test   : (36052, 7225) (36052,)
shape of set2_cv and y_cv       : (24155, 7225) (24155,)
```

2.2.1 Hyper parameter Tuning to find best alpha :: RandomizedSearchCV set2

In [90]:

```
%%time
# Link : https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.Rand

from sklearn.model_selection import RandomizedSearchCV
from sklearn import linear_model
svm = linear_model.SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
alpha = [0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 1000000]
params = {'alpha':[0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 10000]}
clf_2 = RandomizedSearchCV(svm, params,cv=2,scoring='roc_auc', return_train_score=True)
clf_2.fit(set2_train, y_train)

train_auc     = clf_2.cv_results_['mean_train_score']
train_auc_std = clf_2.cv_results_['std_train_score']
cv_auc        = clf_2.cv_results_['mean_test_score']
cv_auc_std    = clf_2.cv_results_['std_test_score']

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

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

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

plt.legend()
plt.grid(True)
plt.xscale('log')
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs alpha : Error Plot")
plt.show()
```



```
CPU times: user 10.1 s, sys: 244 ms, total: 10.4 s
Wall time: 9.92 s
```

In [91]:

```
clf_2.best_params_
```

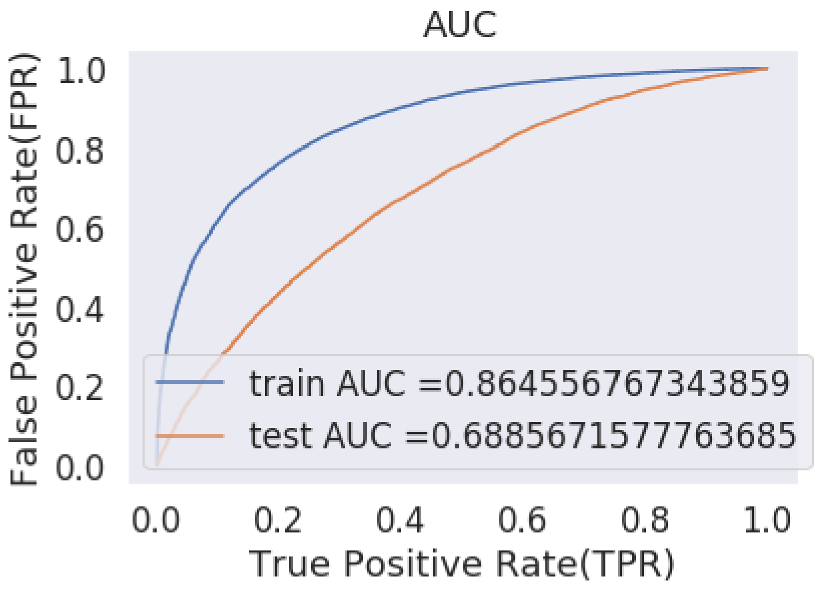
Out[91]:

```
{'alpha': 0.0001}
```

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

In [92]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
# link : https://github.com/scikit-learn/scikit-learn/issues/7278
from sklearn import linear_model
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

clf_2 = linear_model.SGDClassifier(alpha = 0.0001,loss = 'hinge', penalty='l2', class_w
svm = CalibratedClassifierCV(cv=2, method='sigmoid').fit(set2_train, y_train)


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

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

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

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



```
CPU times: user 9.4 s, sys: 1.49 s, total: 10.9 s
Wall time: 8.04 s
```

2.2.3 Confusion Matrix set2_train and set2_test

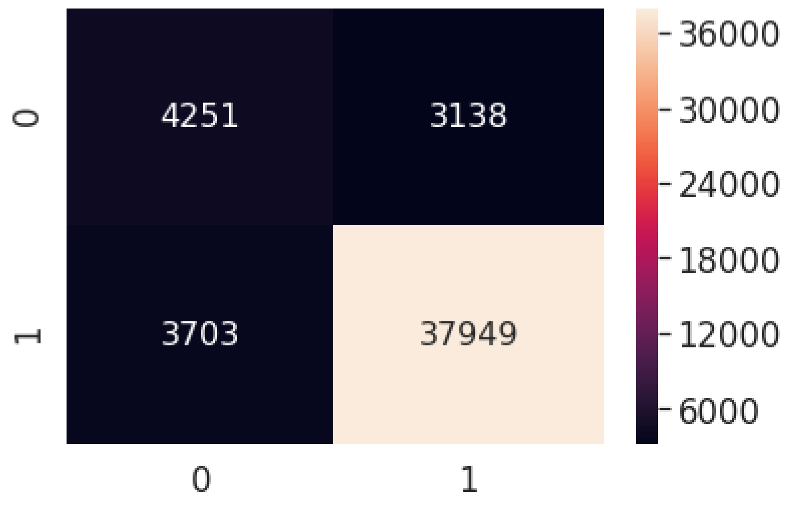
In [93]:

```
#Confustion Matrix Set2_train

svm.fit(set2_train,y_train)
y_train_pred_2 = svm.predict_proba(set2_train)[:,1]
conf_matr_df_train_2 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_2,thr


sns.set(font_scale=1.5)#for label size
sns.heatmap(conf_matr_df_train_2, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999542102075 for threshold 0.792
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa601a898>
```



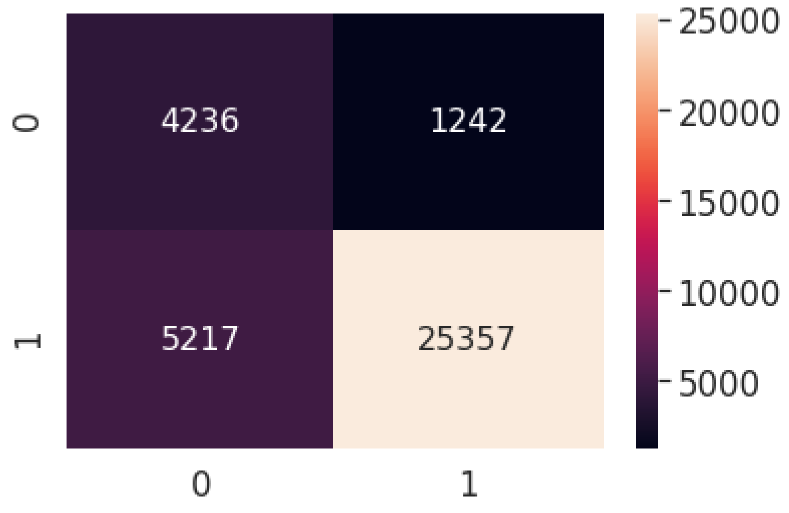
In [94]:

```
# Confusion Matrix set2_test

svm.fit(set2_test, y_test)
y_test_pred_2 = svm.predict_proba(set2_test)[:, 1]
conf_matr_df_test_2 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred_2, thre

sns.set(font_scale=1.5) # label size
sns.heatmap(conf_matr_df_test_2, annot=True, annot_kws={"size":16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.821
```

Out[94]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa5f84cc0>
```



2.3 Apply SVM on set3 AVG W2V

Merging features encoding numerical + categorical features AVG W2V, SET3

In [95]:

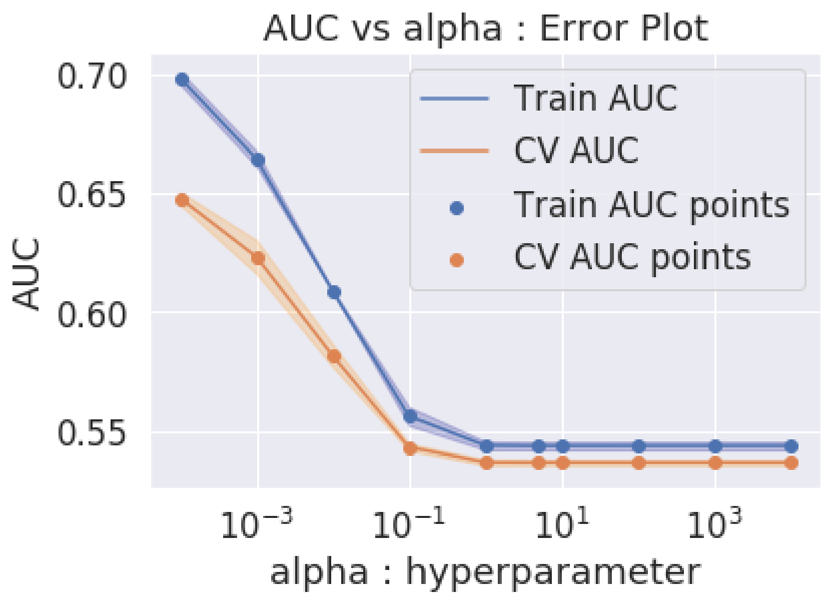
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx
# set3_ = all categorical features + numarical features + essays_avg_w2v_vectors , proj


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

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

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

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

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

2.3.1 Hyperparameter Tuning to find best lalpha :: RandomizedsearchCV set3

In [96]:

```
%%time
# Link : https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.Rand

from sklearn.model_selection import RandomizedSearchCV
from sklearn import linear_model
svm = linear_model.SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
alpha = [0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 10000]
params = {'alpha':[0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 10000]}
clf_3 = RandomizedSearchCV(svm, params,cv=2,scoring='roc_auc', return_train_score=True)
clf_3.fit(set3_train, y_train)

train_auc     = clf_3.cv_results_['mean_train_score']
train_auc_std = clf_3.cv_results_['std_train_score']
cv_auc        = clf_3.cv_results_['mean_test_score']
cv_auc_std    = clf_3.cv_results_['std_test_score']

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

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

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

plt.legend()
plt.grid(True)
plt.xscale('log')
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs alpha : Error Plot")
plt.show()
```



```
CPU times: user 34.3 s, sys: 3.58 s, total: 37.9 s
Wall time: 37.5 s
```

In [97]:

```
clf_3.best_params_
```

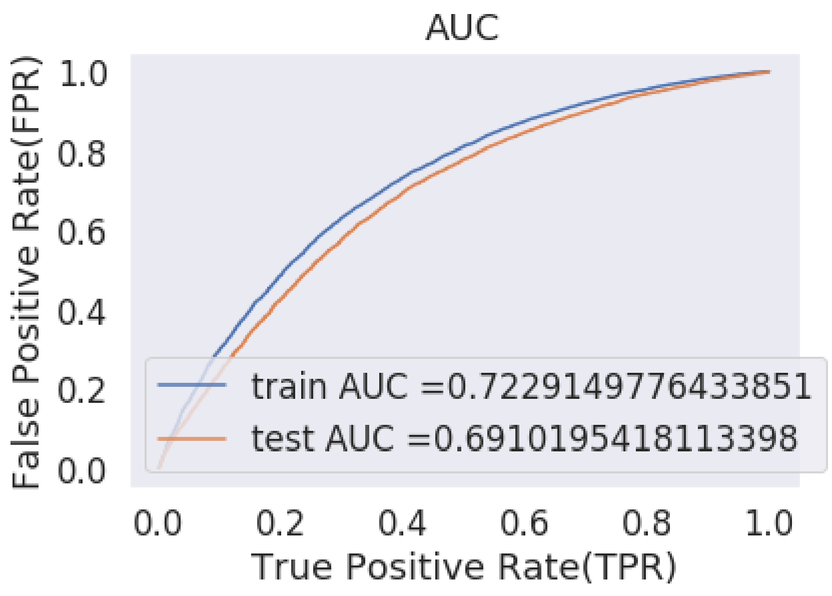
Out[97]:

```
{'alpha': 0.0001}
```

2.3.2 Train model using the best hyper-parameter alpha value : set3

In [98]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
# link : https://github.com/scikit-learn/scikit-learn/issues/7278
from sklearn import linear_model
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

clf_3 = linear_model.SGDClassifier(alpha = 0.0001,loss = 'hinge', penalty='l2', class_w
svm = CalibratedClassifierCV(cv=2, method='sigmoid').fit(set3_train, y_train)


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

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

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

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



```
CPU times: user 2min 5s, sys: 3.2 s, total: 2min 8s
Wall time: 2min 3s
```

2.3.3. Confusion Matrix of set3 _train and test

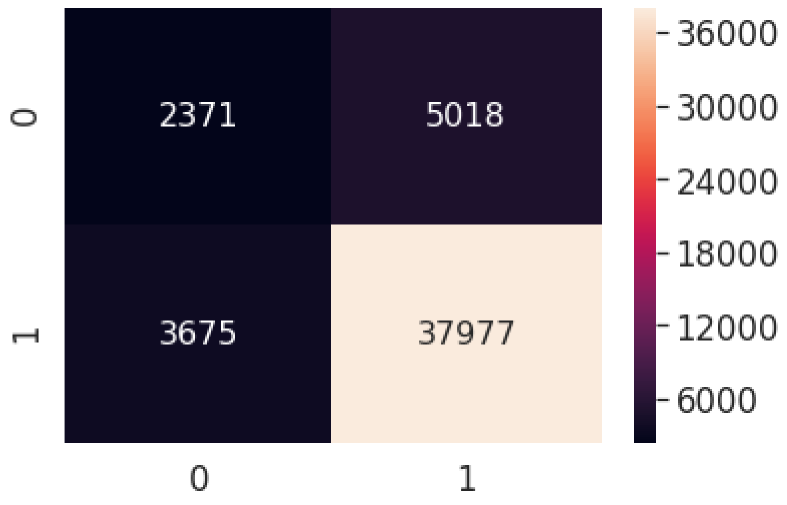
In [99]:

```
#Confustion Matrix Set3_train

svm.fit(set3_train,y_train)
y_train_pred_3 = svm.predict_proba(set3_train)[:,1]
conf_matr_df_train_3 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_3,thr


sns.set(font_scale=1.5)#for label size
sns.heatmap(conf_matr_df_train_3, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999542102075 for threshold 0.776
```

Out[99]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa60dbe10>
```



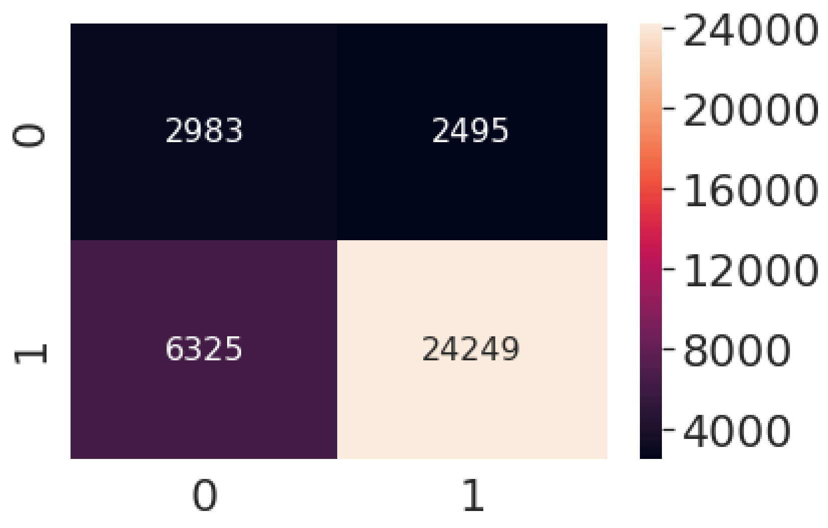
In [100]:

```
# Confusion Matrix set3_test
svm.fit(set3_test, y_test)
y_test_pred_3 = svm.predict_proba(set3_test)[:,1]
conf_matr_df_test_3 = pd.DataFrame(confusion_matrix(y_test, predict(y_test_pred_3, thre

sns.set(font_scale=2) # for label size
sns.heatmap(conf_matr_df_test_3, annot=True, annot_kws={'size': 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.2499999666760907 for threshold 0.818
```

Out[100]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa60820f0>
```



2.4 Applying SVM on TFIDF W2V, SET 4

Merging features encoding numerical + categorical features TFIDF w2V set4

In [101]:

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

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

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

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

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

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

2.4.1.A. Hyperparameter Tuning to find best alpha : SimpleCrossValidation , set4

In [102]:

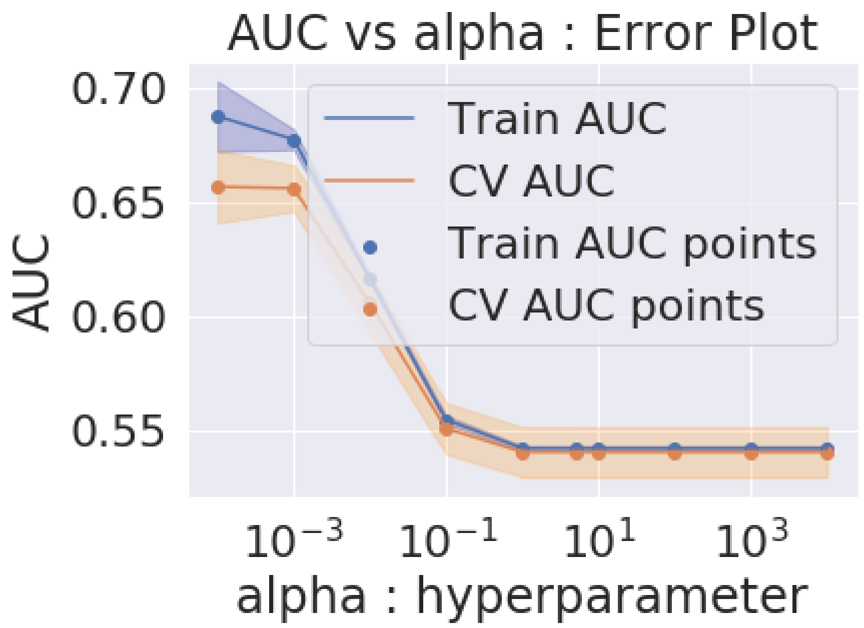
```
%%time
from sklearn import linear_model
from sklearn.metrics import accuracy_score
params = [10**-4, 10**-2, 1, 10, 10**2,  10**4]
for i in params:
    # instantiate learning model (alpha = 1000)
    clf = linear_model.SGDClassifier(loss='hinge', penalty='l2', alpha = i, class_weigh
    # fitting the model on crossvalidation train

    clf.fit(set4_train, y_train)
    # predict the response on the crossvalidation train

    pred = clf.predict(set4_cv)
    # evaluate CV accuracy

    acc = accuracy_score(y_cv, pred, normalize=True) * float(100)
    print('\nCV accuracy for alpha = %d is %d%%' % (i, acc))
```

```
CV accuracy for alpha = 0 is 74%

CV accuracy for alpha = 0 is 63%

CV accuracy for alpha = 1 is 84%

CV accuracy for alpha = 10 is 84%

CV accuracy for alpha = 100 is 84%

CV accuracy for alpha = 10000 is 84%
CPU times: user 12.5 s, sys: 0 ns, total: 12.5 s
Wall time: 12.5 s
```

2.4.1 Hyperparameter Tuning to find best alpha : RandomizedSearchCV , set4

In [103]:

```
%%time
# Link : https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.Rand

from sklearn.model_selection import RandomizedSearchCV
from sklearn import linear_model
svm = linear_model.SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
alpha = [0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 10000]
params = {'alpha':[0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 10000]}
clf_4 = RandomizedSearchCV(svm, params,cv=10,scoring='roc_auc', return_train_score=True
clf_4.fit(set4_train, y_train)

train_auc     = clf_4.cv_results_['mean_train_score']
train_auc_std = clf_4.cv_results_['std_train_score']
cv_auc        = clf_4.cv_results_['mean_test_score']
cv_auc_std    = clf_4.cv_results_['std_test_score']

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

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

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

plt.legend()
plt.grid(True)
plt.xscale('log')
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs alpha : Error Plot")
plt.show()
```



```
CPU times: user 3min 45s, sys: 15.4 s, total: 4min
```

```
Wall time: 4min
```

In [104]:

```
clf_4.best_params_
```

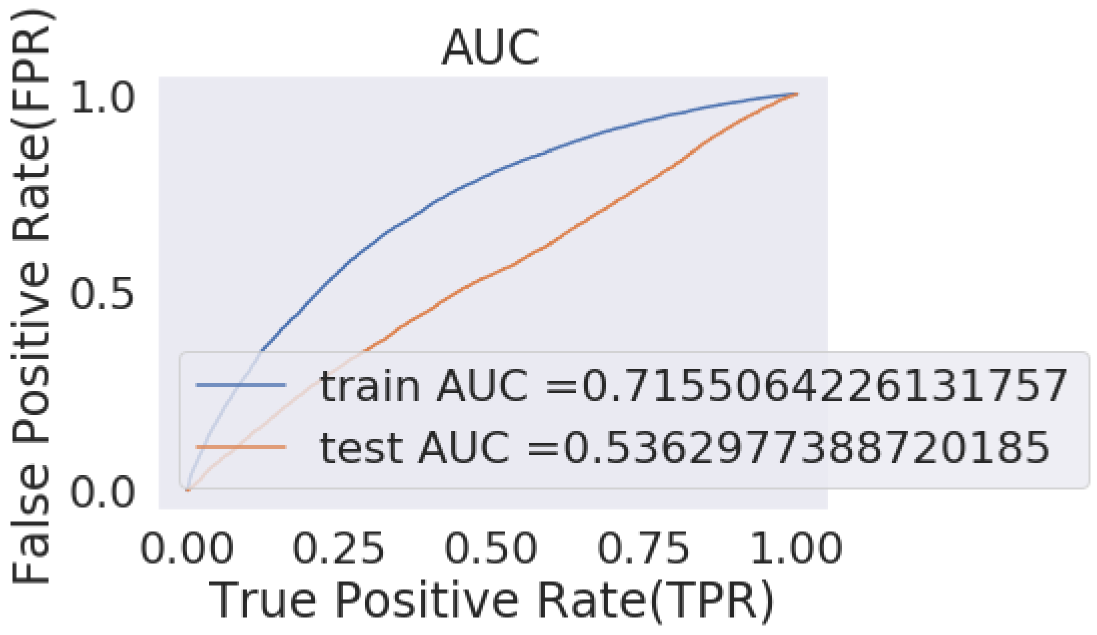
Out[104]:

```
{'alpha': 0.0001}
```

2.4.2 Train model using best hyperparameter alpha value : set4

In [105]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
# link : https://github.com/scikit-learn/scikit-learn/issues/7278
from sklearn import linear_model
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

clf_4 = linear_model.SGDClassifier(alpha = 0.0001,loss = 'hinge', penalty='l2',class_we
svm = CalibratedClassifierCV(cv=2, method='sigmoid').fit(set4_train, y_train)


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

train_fpr, train_tpr, thresholds = roc_curve(y_train, svm.predict_proba(set4_train)[:,1
test_fpr, test_tpr , thresholds = roc_curve(y_test, svm.predict_proba(set4_test)[:,1])

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

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



```
CPU times: user 1min 59s, sys: 1.81 s, total: 2min 1s
Wall time: 1min 59s
```

2.4.3 Confusion Matrix set4_train and set4_test

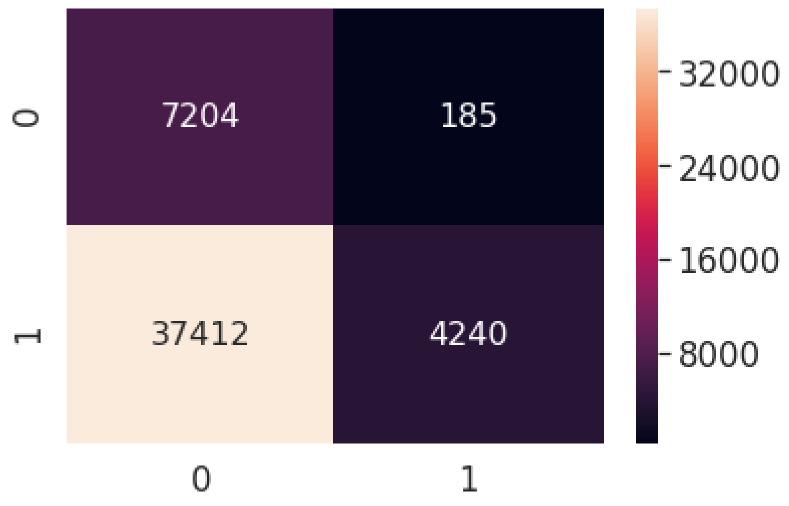
In [106]:

```
# confusion Matrix set4_train

svm.fit(set4_train,y_train)
y_train_pred_4 = svm.predict_proba(set4_train)[:,1]
conf_matr_df_train_4 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_4,thr


sns.set(font_scale=1.5)#for label size
sns.heatmap(conf_matr_df_train_4, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999542102075 for threshold 0.918
```

Out[106]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa615e4a8>
```



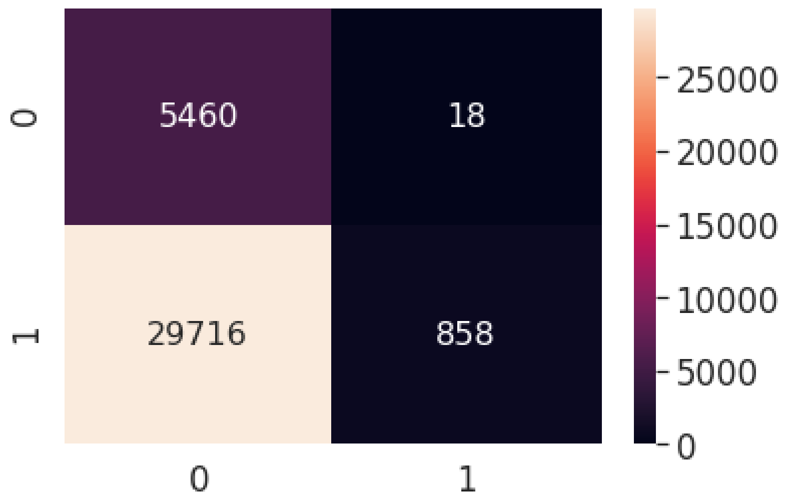
In [107]:

```
#Confustion Matrix Set4_test

svm.fit(set4_test,y_test)
y_test_pred_4 = svm.predict_proba(set4_test)[:,1]
conf_matr_df_test_4 = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred_4,thresh


sns.set(font_scale=1.5)#for label size
sns.heatmap(conf_matr_df_test_4, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.942
```

Out[107]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa5f920f0>
```

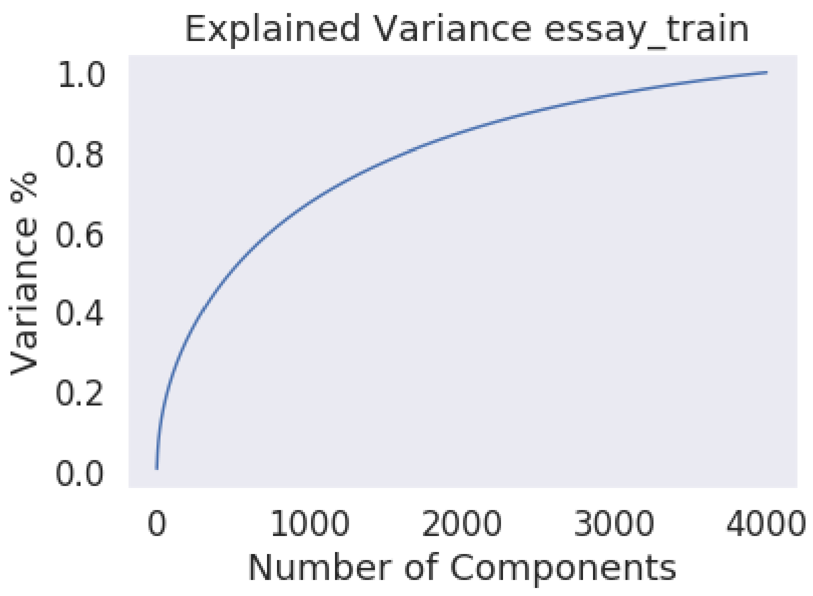


2.5 Appling Support Vector Machines on different kind of featurization as mentioned in the instructions set5

Performing Elbow method to find the best number of Components

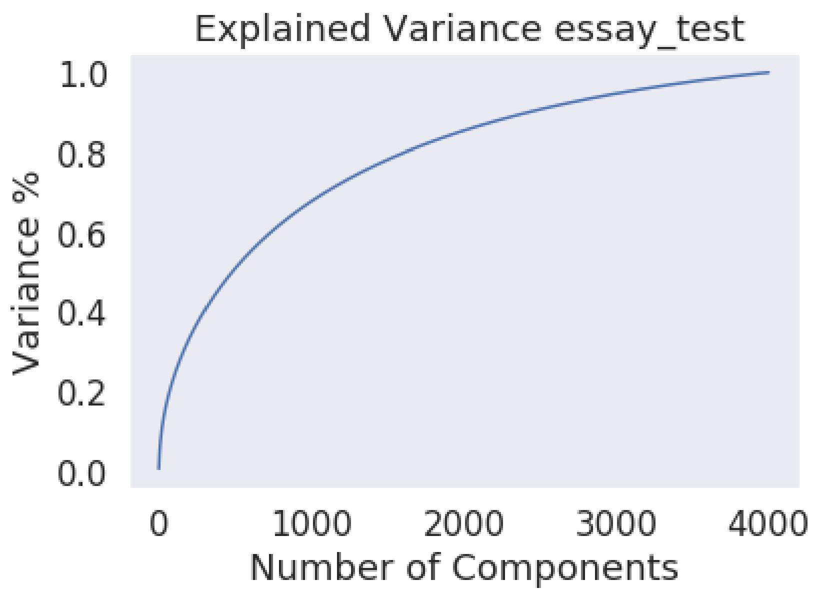
In [108]:

```
from sklearn.decomposition import TruncatedSVD
import matplotlib.pyplot as plt

tsvd = TruncatedSVD(n_components=4000)
svd_train = tsvd.fit(essays_tfidf_train)

percentage_var_explained_train = svd_train.explained_variance_ /np.sum(svd_train.explai
cum_var_explained_train = np.cumsum(percentage_var_explained_train)

plt.figure()
plt.plot(cum_var_explained_train)
plt.xlabel('Number of Components')
plt.ylabel('Variance %') #for each component
plt.title(' Explained Variance essay_train')
plt.grid()
plt.show()
```



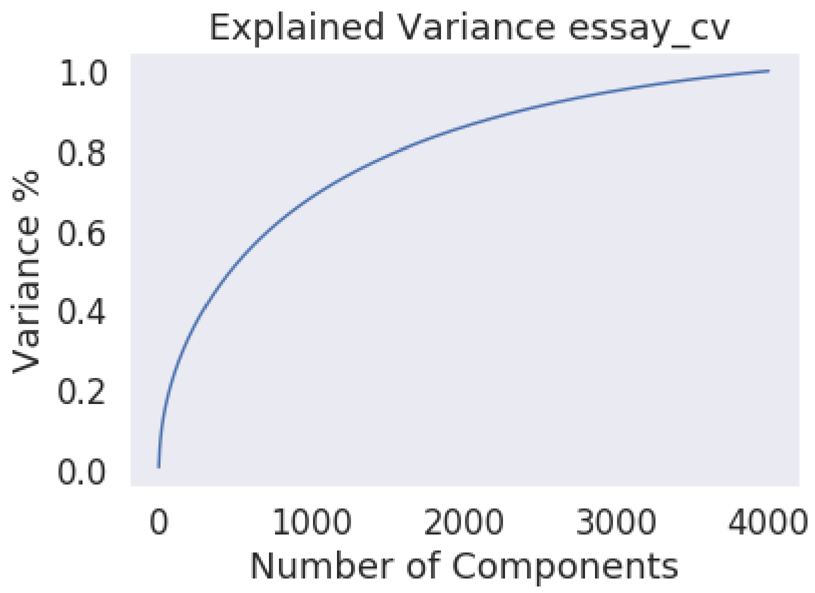
In [109]:

```
from sklearn.decomposition import TruncatedSVD
import matplotlib.pyplot as plt

tsvd = TruncatedSVD(n_components=4000)
svd_test = tsvd.fit(essays_tfidf_test)

percentage_var_explained_test = svd_test.explained_variance_ /np.sum(svd_test.explained_
cum_var_explained_test = np.cumsum(percentage_var_explained_test)

plt.figure()
plt.plot(cum_var_explained_test)
plt.xlabel('Number of Components')
plt.ylabel('Variance %') #for each component
plt.title(' Explained Variance essay_test')
plt.grid()
plt.show()
```



In [110]:

```
from sklearn.decomposition import TruncatedSVD
import matplotlib.pyplot as plt

tsvd = TruncatedSVD(n_components=4000)
svd_cv = tsvd.fit(essays_tfidf_cv)

percentage_var_explained_cv = svd_cv.explained_variance_ /np.sum(svd_cv.explained_varia
cum_var_explained_cv = np.cumsum(percentage_var_explained_cv)

plt.figure()
plt.plot(cum_var_explained_cv)
plt.xlabel('Number of Components')
plt.ylabel('Variance %') #for each component
plt.title(' Explained Variance essay_cv')
plt.grid()
plt.show()
```



In [111]:

```
from sklearn.decomposition import TruncatedSVD
from sklearn.random_projection import sparse_random_matrix

svd = TruncatedSVD(n_components= 4000, n_iter=7, random_state=42)
svd.fit(essays_tfidf_train)
svd_essay_train = svd.transform(essays_tfidf_train)
```

In [112]:

```
#essay_test
svd = TruncatedSVD(n_components= 4000, n_iter=7, random_state=42)
svd.fit(essays_tfidf_test)
svd_essay_test = svd.transform(essays_tfidf_test)
```

In [113]:

```
#essay_cv
svd = TruncatedSVD(n_components= 4000, n_iter=7, random_state=42)
svd.fit(essays_tfidf_cv)
svd_essay_cv = svd.transform(essays_tfidf_cv)
```

Merging All categories , Numarical and selected features

In [114]:

```
# Merging all categories , Numarical and selectes features
# set5 = all categories + Numarical + sse essay + svd essay

set5_train = hstack((clean_categories_one_hot_train, clean_subcategories_one_hot_train,
                     teacher_prefix_one_hot_train,project_grade_category_one_hot_train,
                     teacher_number_of_previously_posted_projects_normalizer_train, sse_


set5_test = hstack((clean_categories_one_hot_test, clean_subcategories_one_hot_test, sc
                     teacher_prefix_one_hot_test,project_grade_category_one_hot_test, p
                     teacher_number_of_previously_posted_projects_normalizer_test, sse_

set5_cv   = hstack((clean_categories_one_hot_cv, clean_subcategories_one_hot_cv, school_
                     teacher_prefix_one_hot_cv,project_grade_category_one_hot_cv, price
                     teacher_number_of_previously_posted_projects_normalizer_cv, sse_cv

print("Final Matrix of set5 Shape")
print("Shape of set5_train and y_train :", set5_train.shape, y_train.shape)
print("Shape of set5_test and y_test   :", set5_test.shape , y_test.shape)
print("Shape of set5_cv and y_cv       :", set5_cv.shape , y_cv.shape)
```

```
Final Matrix of set5 Shape
Shape of set5_train and y_train : (49041, 4102) (49041,)
Shape of set5_test and y_test   : (36052, 4102) (36052,)
Shape of set5_cv and y_cv       : (24155, 4102) (24155,)
```

2.5.1.A. Hyperparameter Tuning to find best alpha : SimpleCrossValidation , set5

In [115]:

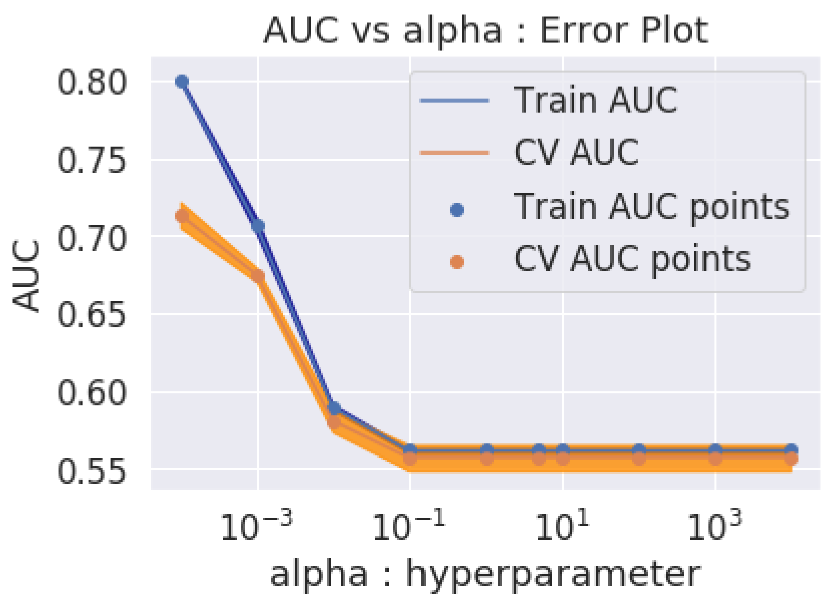
```
%%time
from sklearn import linear_model
from sklearn.metrics import accuracy_score
params = [10**-4, 10**-2, 1, 10, 10**2,  10**4]
for i in params:
    # instantiate learning model (alpha = 1000)
    clf = linear_model.SGDClassifier(loss='hinge', penalty='l2', alpha = i, class_weigh
    # fitting the model on crossvalidation train

    clf.fit(set5_train, y_train)
    # predict the response on the crossvalidation train

    pred = clf.predict(set5_cv)
    # evaluate CV accuracy

    acc = accuracy_score(y_cv, pred, normalize=True) * float(100)
    print('\nCV accuracy for alpha = %d is %d%%' % (i, acc))
```

```
CV accuracy for alpha = 0 is 65%

CV accuracy for alpha = 0 is 53%

CV accuracy for alpha = 1 is 66%

CV accuracy for alpha = 10 is 53%

CV accuracy for alpha = 100 is 84%

CV accuracy for alpha = 10000 is 84%
CPU times: user 44 s, sys: 0 ns, total: 44 s
Wall time: 44 s
```

2.5.1.B Hyperparameter Tuning to find best alpha : RandomSearchCV , set5

In [116]:

```
%%time
# Link : https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.Rand

from sklearn.model_selection import RandomizedSearchCV
from sklearn import linear_model
svm = linear_model.SGDClassifier(loss='hinge', penalty='l2', class_weight='balanced')
alpha = [0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 10000]
params = {'alpha':[0.0001, 0.001, 0.01, 0.1, 1, 5, 10, 100, 1000, 10000]}
clf_5 = RandomizedSearchCV(svm, params,cv=5,scoring='roc_auc', return_train_score=True)
clf_5.fit(set5_train, y_train)

train_auc     = clf_5.cv_results_['mean_train_score']
train_auc_std = clf_5.cv_results_['std_train_score']
cv_auc        = clf_5.cv_results_['mean_test_score']
cv_auc_std    = clf_5.cv_results_['std_test_score']

plt.plot(params['alpha'], train_auc, label='Train AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(params['alpha'],train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.8,color='darkblue')

plt.plot(params['alpha'], cv_auc, label='CV AUC')
# this code is copied from here: https://stackoverflow.com/a/48803361/4084039
plt.gca().fill_between(params['alpha'],cv_auc - cv_auc_std,cv_auc +
cv_auc_std,alpha=0.8,color='darkorange')

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

plt.legend()
plt.grid(True)
plt.xscale('log')
plt.xlabel("alpha : hyperparameter")
plt.ylabel("AUC")
plt.title("AUC vs alpha : Error Plot")
plt.show()
```



```
CPU times: user 6min 33s, sys: 1min 18s, total: 7min 51s
Wall time: 7min 51s
```

In [117]:

```
clf_5.best_params_
```

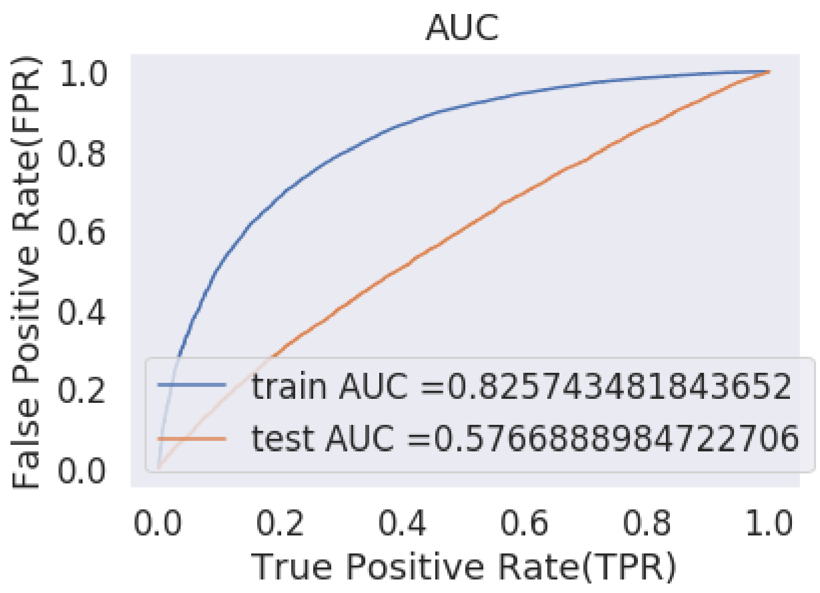
Out[117]:

```
{'alpha': 0.0001}
```

2.5.2 train model by using best hyperparameter value alpha : set5

In [118]:

```
%%time
# https://scikitlearn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklea
# link : https://github.com/scikit-learn/scikit-learn/issues/7278
from sklearn import linear_model
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

clf_5 = linear_model.SGDClassifier(alpha =0.0001,loss = 'hinge', penalty='l2',class_wei
svm = CalibratedClassifierCV(cv=5, method='sigmoid').fit(set5_train, y_train)


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

train_fpr, train_tpr, thresholds = roc_curve(y_train, svm.predict_proba(set5_train)[:,1
test_fpr, test_tpr , thresholds = roc_curve(y_test, svm.predict_proba(set5_test)[:,1])

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

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



```
CPU times: user 8min 23s, sys: 13.5 s, total: 8min 36s
Wall time: 8min 36s
```

2.5.3 Confusion Matrix set5_train, test,

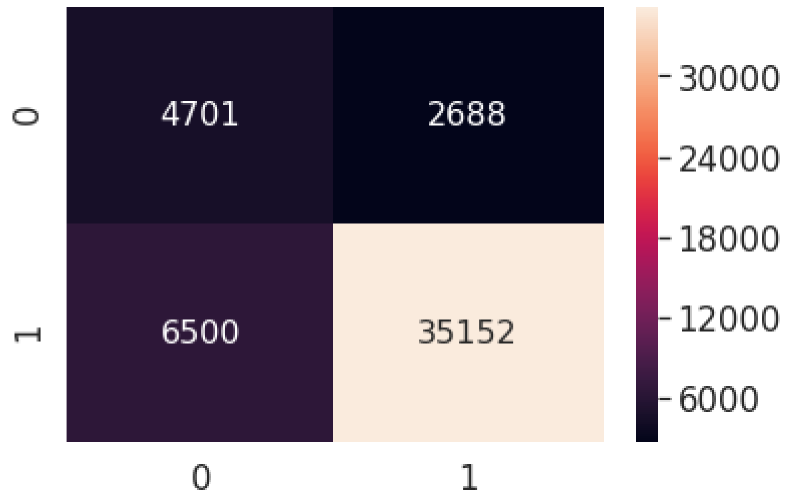
In [119]:

```
#Confustion Matrix Set5_train

svm.fit(set5_train,y_train)
y_train_pred_5 = svm.predict_proba(set5_train)[:,1]
conf_matr_df_train_5 = pd.DataFrame(confusion_matrix(y_train,predict(y_train_pred_5,thr


sns.set(font_scale=1.5)#for label size
sns.heatmap(conf_matr_df_train_5, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.24999999542102075 for threshold 0.801
```

Out[119]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa5ef8240>
```



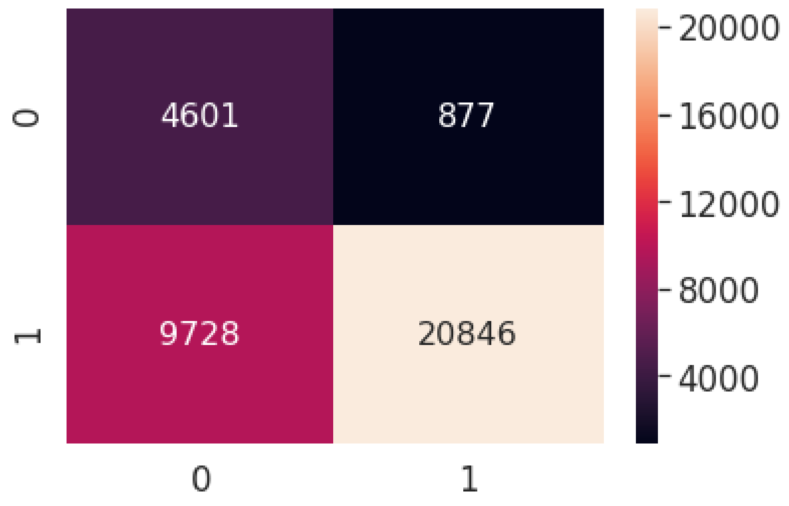
In [120]:

```
#Confustion Matrix Set5_test

svm.fit(set5_test,y_test)
y_test_pred_5 = svm.predict_proba(set5_test)[:,1]
conf_matr_df_test_5 = pd.DataFrame(confusion_matrix(y_test,predict(y_test_pred_5,thresh


sns.set(font_scale=1.5)#for label size
sns.heatmap(conf_matr_df_test_5, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.25 for threshold 0.847
```

Out[120]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbaa5e5d5f8>
```



3. Conclusion

In [1]:

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

p.field_names = ["Vectorizer", "Model", "Hyper parameter(alpha)", "AUC"]

p.add_row(["BOW","SVM",0.1, 0.65])
p.add_row(["TFIDF","SVM",0.0001, 0.68])
p.add_row(["AVG W2V","SVM",0.0001, 0.69])
p.add_row(["TFIDF W2V","SVM",0.0001, 0.53])
p.add_row(["Added features SET5","SVM",0.0001, 0.57])

print(p)
```

```
+---------------------+-------+------------------------+------+
|      Vectorizer     | Model | Hyper parameter(alpha) | AUC  |
+---------------------+-------+------------------------+------+
|         BOW         |  SVM  |          0.1           | 0.65 |
|        TFIDF        |  SVM  |         0.0001         | 0.68 |
|       AVG W2V       |  SVM  |         0.0001         | 0.69 |
|      TFIDF W2V      |  SVM  |         0.0001         | 0.53 |
| Added features SET5 |  SVM  |         0.0001         | 0.57 |
+---------------------+-------+------------------------+------+
```

Observation :

1. Highest AUC score is 0.69

2. Without text data also AUC score is greater than 0.5

3. SVM model is good because for every vectorizer AUC score is greater than 0.5

Thank You.

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