DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`

`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

Feature	Description
`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.

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os


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.sha
pe)
print('-'*50)
print("The attributes of data :", project_data.columns.values
)
```

```
Number of data points in train data (109248, 1
7)
----------------------------------------------
----
The attributes of data : ['Unnamed: 0' 'id' 't
eacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_c
ategory'
 'project_subject_categories' 'project_subject
_subcategories'
 'project_title' 'project_essay_1' 'project_es
say_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects
' 'project_is_approved']
```

In [4]:

```
print("Number of data points in train data", resource_data.sh
ape)
```

```
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[4]:

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

1.2 preprocessing of `project_subject_categories`

In [5]:

```
catogories = list(project_data['project_subject_categories'].
values)
# remove special characters from list of strings python: http
s://stackoverflow.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-pyth
on/
# https://stackoverflow.com/questions/23669024/how-to-strip-a
-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whit
espace-in-a-string-in-python
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", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the
catogory based on space "Math & Science"=> "Math","&", "Scien
ce"
            j=j.replace('The','') # if we have the words "The
" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(s
pace) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc
", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the &
 value into
```

```
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [6]:

```
sub_catogories = list(project_data['project_subject_subcatego
ries'].values)
# remove special characters from list of strings python: http
s://stackoverflow.com/a/47301924/4084039

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

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", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the
catogory based on space "Math & Science"=> "Math","&", "Scien
ce"
            j=j.replace('The','') # if we have the words "The
" we are going to replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(s
pace) with ''(empty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc
", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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

# count of all the words in corpus python: https://stackoverf
low.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=l
ambda kv: kv[1]))
```

1.3 preprocessing of `school_state`

In [7]:

```
school_states = list(project_data['school_state'].values)
# remove special characters from list of strings python: http
s://stackoverflow.com/a/47301924/4084039

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

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

```
project_data.drop(['school_state'], axis=1, inplace=True)
project_data['school_state'] = school_states_list



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

school_states_dict = dict(my_counter)
sorted_school_states_dict = dict(sorted(school_states_dict.it
ems(), key=lambda kv: kv[1]))
```

1.3 preprocessing of `teacher_prefix`

In [8]:

```
#NaN values in techer prefix will create a problem while enco
ding,so we replace NaN values with the mode of that particula
r column
#removing dot(.) since it is a special character
mode_of_teacher_prefix = project_data['teacher_prefix'].value
_counts().index[0]


project_data['teacher_prefix'] = project_data['teacher_prefix
'].fillna(mode_of_teacher_prefix)
```

In [9]:

```
prefixes = []

for i in range(len(project_data)):
    a = project_data["teacher_prefix"][i].replace(".", "")
    prefixes.append(a)
```

In [10]:

```
project_data.drop(['teacher_prefix'], axis = 1, inplace = Tru
e)
project_data["teacher_prefix"] = prefixes
print("After removing the special characters ,Column values:
 ")
np.unique(project_data["teacher_prefix"].values)
```

```
After removing the special characters ,Column
values:
```

Out[10]:

```
array(['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher'], dt
ype=object)
```

1.3 preprocessing of `project_grade_category`

In [11]:

```
# We need to get rid of The spaces between the text and the h
yphens because they're special characters.
#Rmoving multiple characters from a string in Python
#https://stackoverflow.com/questions/3411771/multiple-charact
er-replace-with-python

project_grade_category = []

for i in range(len(project_data)):
    a = project_data["project_grade_category"][i].replace(" "
, "_").replace("-", "_")
    project_grade_category.append(a)
```

In [12]:

```
project_data.drop(['project_grade_category'], axis = 1, inpla
ce = True)
project_data["project_grade_category"] = project_grade_catego
ry
print("After removing the special characters ,Column values:
 ")
np.unique(project_data["project_grade_category"].values)
```

```
After removing the special characters ,Column
values:
```

Out[12]:

```
array(['Grades_3_5', 'Grades_6_8', 'Grades_9_1
2', 'Grades_PreK_2'], dtype=object)
```

1.3 Text preprocessing

In [13]:

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

In [14]:

```
project_data.head(2)
```

Out[14]:

	Unnamed: 0	id	teacher_id
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc
1	140945	p258326	897464ce9ddc600bced1151f324dd63a

In [15]:

```python
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are work
ing on English as their second or third langua
ges. We are a melting pot of refugees, immigra
nts, and native-born Americans bringing the gi
ft of language to our school. \r\n\r\n We have
 over 24 languages represented in our English
Learner program with students at every level o
f mastery.  We also have over 40 countries rep
resented with the families within our school.
 Each student brings a wealth of knowledge and
 experiences to us that open our eyes to new c
ultures, beliefs, and respect.\"The limits of
your language are the limits of your world.\"-
Ludwig Wittgenstein  Our English learner's hav
e a strong support system at home that begs fo
r more resources.  Many times our parents are
learning to read and speak English along side
of their children.  Sometimes this creates bar
riers for parents to be able to help their chi
ld learn phonetics, letter recognition, and ot
her reading skills.\r\n\r\nBy providing these
dvd's and players, students are able to contin
ue their mastery of the English language even
```

```
if no one at home is able to assist.  All fami
lies with students within the Level 1 proficie
ncy status, will be a offered to be a part of 
this program.  These educational videos will b
e specially chosen by the English Learner Teac
her and will be sent home regularly to watch. 
 The videos are to help the child develop earl
y reading skills.\r\n\r\nParents that do not h
ave access to a dvd player will have the oppor
tunity to check out a dvd player to use for th
e year.  The plan is to use these videos and e
ducational dvd's for the years to come for oth
er EL students.\r\nnannan
==============================================
====
The 51 fifth grade students that will cycle th
rough my classroom this year all love learning
, at least most of the time. At our school, 97
.3% of the students receive free or reduced pr
ice lunch. Of the 560 students, 97.3% are mino
rity students. \r\nThe school has a vibrant co
mmunity that loves to get together and celebra
te. Around Halloween there is a whole school p
arade to show off the beautiful costumes that 
students wear. On Cinco de Mayo we put on a bi
g festival with crafts made by the students, d
ances, and games. At the end of the year the s
chool hosts a carnival to celebrate the hard w
ork put in during the school year, with a dunk
 tank being the most popular activity.My stude
nts will use these five brightly colored Hokki
 stools in place of regular, stationary, 4-leg
ged chairs. As I will only have a total of ten
 in the classroom and not enough for each stud
ent to have an individual one, they will be us
ed in a variety of ways. During independent re
ading time they will be used as special chairs
```

```
 students will each use on occasion. I will ut
ilize them in place of chairs at my small grou
p tables during math and reading times. The re
st of the day they will be used by the student
s who need the highest amount of movement in t
heir life in order to stay focused on school.\
r\n\r\nWhenever asked what the classroom is mi
ssing, my students always say more Hokki Stool
s. They can't get their fill of the 5 stools w
e already have. When the students are sitting 
in group with me on the Hokki Stools, they are
 always moving, but at the same time doing the
ir work. Anytime the students get to pick wher
e they can sit, the Hokki Stools are the first
 to be taken. There are always students who he
ad over to the kidney table to get one of the 
stools who are disappointed as there are not e
nough of them. \r\n\r\nWe ask a lot of student
s to sit for 7 hours a day. The Hokki stools w
ill be a compromise that allow my students to 
do desk work and move at the same time. These 
stools will help students to meet their 60 min
utes a day of movement by allowing them to act
ivate their core muscles for balance while the
y sit. For many of my students, these chairs w
ill take away the barrier that exists in schoo
ls for a child who can't sit still.nannan
==============================================
====
How do you remember your days of school? Was i
t in a sterile environment with plain walls, r
ows of desks, and a teacher in front of the ro
om? A typical day in our room is nothing like 
that. I work hard to create a warm inviting th
emed room for my students look forward to comi
ng to each day.\r\n\r\nMy class is made up of 
28 wonderfully unique boys and girls of mixed
```

```
races in Arkansas.\r\nThey attend a Title I sc
hool, which means there is a high enough perce
ntage of free and reduced-price lunch to quali
fy. Our school is an \"open classroom\" concep
t, which is very unique as there are no walls 
separating the classrooms. These 9 and 10 year
-old students are very eager learners; they ar
e like sponges, absorbing all the information 
and experiences and keep on wanting more.With 
these resources such as the comfy red throw pi
llows and the whimsical nautical hanging decor
 and the blue fish nets, I will be able to hel
p create the mood in our classroom setting to 
be one of a themed nautical environment. Creat
ing a classroom environment is very important 
in the success in each and every child's educa
tion. The nautical photo props will be used wi
th each child as they step foot into our class
room for the first time on Meet the Teacher ev
ening. I'll take pictures of each child with t
hem, have them developed, and then hung in our
 classroom ready for their first day of 4th gr
ade.  This kind gesture will set the tone befo
re even the first day of school! The nautical 
thank you cards will be used throughout the ye
ar by the students as they create thank you ca
rds to their team groups.\r\n\r\nYour generous
 donations will help me to help make our class
room a fun, inviting, learning environment fro
m day one.\r\n\r\nIt costs lost of money out o
f my own pocket on resources to get our classr
oom ready. Please consider helping with this p
roject to make our new school year a very succ
essful one. Thank you!nannan
==============================================
====
My kindergarten students have varied disabilit
```

ies ranging from speech and language delays, c
ognitive delays, gross/fine motor delays, to a
utism. They are eager beavers and always striv
e to work their hardest working past their lim
itations. \r\n\r\nThe materials we have are th
e ones I seek out for my students. I teach in
a Title I school where most of the students re
ceive free or reduced price lunch. Despite th
eir disabilities and limitations, my students
love coming to school and come eager to learn
and explore.Have you ever felt like you had an
ts in your pants and you needed to groove and
move as you were in a meeting? This is how my
kids feel all the time. The want to be able to
 move as they learn or so they say.Wobble chai
rs are the answer and I love then because they
 develop their core, which enhances gross moto
r and in Turn fine motor skills. \r\nThey also
 want to learn through games, my kids don't wa
nt to sit and do worksheets. They want to lear
n to count by jumping and playing. Physical en
gagement is the key to our success. The number
 toss and color and shape mats can make that h
appen. My students will forget they are doing
work and just have the fun a 6 year old deserv
es.nannan
==
====
The mediocre teacher tells. The good teacher e
xplains. The superior teacher demonstrates. Th
e great teacher inspires. -William A. Ward\r\n
\r\nMy school has 803 students which is makeup
 is 97.6% African-American, making up the larg
est segment of the student body. A typical sch
ool in Dallas is made up of 23.2% African-Amer
ican students. Most of the students are on fre
e or reduced lunch. We aren't receiving doctor

```
s, lawyers, or engineers children from rich ba
ckgrounds or neighborhoods. As an educator I a
m inspiring minds of young children and we foc
us not only on academics but one smart, effect
ive, efficient, and disciplined students with
good character.In our classroom we can utilize
 the Bluetooth for swift transitions during cl
ass. I use a speaker which doesn't amplify the
 sound enough to receive the message. Due to t
he volume of my speaker my students can't hear
 videos or books clearly and it isn't making t
he lessons as meaningful. But with the bluetoo
th speaker my students will be able to hear an
d I can stop, pause and replay it at any time.
\r\nThe cart will allow me to have more room f
or storage of things that are needed for the d
ay and has an extra part to it I can use.  The
 table top chart has all of the letter, words
and pictures for students to learn about diffe
rent letters and it is more accessible.nannan
==============================================
====
```

In [16]:

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

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

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

In [17]:

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

```
My kindergarten students have varied disabilit
ies ranging from speech and language delays, c
ognitive delays, gross/fine motor delays, to a
utism. They are eager beavers and always striv
e to work their hardest working past their lim
itations. \r\n\r\nThe materials we have are th
e ones I seek out for my students. I teach in
a Title I school where most of the students re
ceive free or reduced price lunch.  Despite th
eir disabilities and limitations, my students
love coming to school and come eager to learn
and explore.Have you ever felt like you had an
ts in your pants and you needed to groove and
move as you were in a meeting? This is how my
kids feel all the time. The want to be able to
 move as they learn or so they say.Wobble chai
rs are the answer and I love then because they
 develop their core, which enhances gross moto
r and in Turn fine motor skills. \r\nThey also
 want to learn through games, my kids do not w
ant to sit and do worksheets. They want to lea
rn to count by jumping and playing. Physical e
ngagement is the key to our success. The numbe
r toss and color and shape mats can make that
happen. My students will forget they are doing
```

```
 work and just have the fun a 6 year old deser
ves.nannan
==============================================
====
```

In [18]:

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

```
My kindergarten students have varied disabilit
ies ranging from speech and language delays, c
ognitive delays, gross/fine motor delays, to a
utism. They are eager beavers and always striv
e to work their hardest working past their lim
itations.     The materials we have are the on
es I seek out for my students. I teach in a Ti
tle I school where most of the students receiv
e free or reduced price lunch.  Despite their 
disabilities and limitations, my students love
 coming to school and come eager to learn and 
explore.Have you ever felt like you had ants i
n your pants and you needed to groove and move
 as you were in a meeting? This is how my kids
 feel all the time. The want to be able to mov
e as they learn or so they say.Wobble chairs a
re the answer and I love then because they dev
elop their core, which enhances gross motor an
d in Turn fine motor skills.   They also want 
to learn through games, my kids do not want to
 sit and do worksheets. They want to learn to 
count by jumping and playing. Physical engagem
ent is the key to our success. The number toss
 and color and shape mats can make that happen
```

```
. My students will forget they are doing work
and just have the fun a 6 year old deserves.na
nnan
```

In [19]:

```
#remove spacial character: https://stackoverflow.com/a/584354
7/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilit
ies ranging from speech and language delays co
gnitive delays gross fine motor delays to auti
sm They are eager beavers and always strive to
 work their hardest working past their limitat
ions The materials we have are the ones I seek
 out for my students I teach in a Title I scho
ol where most of the students receive free or
reduced price lunch Despite their disabilities
 and limitations my students love coming to sc
hool and come eager to learn and explore Have
you ever felt like you had ants in your pants
and you needed to groove and move as you were
in a meeting This is how my kids feel all the
time The want to be able to move as they learn
 or so they say Wobble chairs are the answer a
nd I love then because they develop their core
 which enhances gross motor and in Turn fine m
otor skills They also want to learn through ga
mes my kids do not want to sit and do workshee
ts They want to learn to count by jumping and
playing Physical engagement is the key to our
success The number toss and color and shape ma
ts can make that happen My students will forge
t they are doing work and just have the fun a
6 year old deserves nannan
```

In [20]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', '
nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', '
ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', '
yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "
it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', '
whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', '
being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', '
if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'b
etween', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in
', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where',
 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', '
same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't",
'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "co
uldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", '
isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't",
'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",
 \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [21]:

```python
# 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 = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopw
ords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:45<00:00, 2
403.41it/s]
```

In [22]:

```python
# after preprocesing
preprocessed_essays[20000]
```

Out[22]:

```
'my kindergarten students varied disabilities
ranging speech language delays cognitive delay
s gross fine motor delays autism they eager be
avers always strive work hardest working past
limitations the materials ones i seek students
 i teach title i school students receive free
reduced price lunch despite disabilities limit
ations students love coming school come eager
learn explore have ever felt like ants pants n
eeded groove move meeting this kids feel time
the want able move learn say wobble chairs ans
wer i love develop core enhances gross motor t
urn fine motor skills they also want learn gam
```

```
es kids not want sit worksheets they want lear
n count jumping playing physical engagement ke
y success the number toss color shape mats mak
e happen my students forget work fun 6 year ol
d deserves nannan'
```

In [23]:

```
#creating a new column with the preprocessed essays and repla
cing it with the original columns
project_data['preprocessed_essays'] = preprocessed_essays
project_data.drop(['project_essay_1'], axis=1, inplace=True)
project_data.drop(['project_essay_2'], axis=1, inplace=True)
project_data.drop(['project_essay_3'], axis=1, inplace=True)
project_data.drop(['project_essay_4'], axis=1, inplace=True)
```

In [24]:

```
essay_word_count=[]
for i in range(len(project_data['preprocessed_essays'])):
    essay_word_count.append(len(project_data['preprocessed_es
says'][i].split()))
```

In [25]:

```
len(project_data['preprocessed_essays'][1].split())
```

Out[25]:

```
109
```

In [26]:

```
essay_word_count[1]
```

Out[26]:

```
109
```

In [27]:

```
project_data['essay_word_count'] = essay_word_count
```

In [28]:

```
#simple example on how to use nltk's sentiment intensity anal
yzer
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
#if it shows an error,then do the following:
# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr
seuss i teach the smallest students with the biggest enthusia
sm \
for learning my students learn in many different ways using a
ll of our senses and multiple intelligences i use a wide rang
e\
of techniques to help all my students succeed students in my
class come from a variety of different backgrounds which make
s\
for wonderful sharing of experiences and cultures including n
ative americans our school is a caring community of successfu
l \
learners which can be seen through collaborative student proj
ect based learning in and out of the classroom kindergartener
s \
in my class love to work with hands on materials and have man
y different opportunities to practice a skill before it is\
mastered having the social skills to work cooperatively with
friends is a crucial aspect of the kindergarten curriculum\
montana is the perfect place to learn about agriculture and n
utrition my students love to role play in our pretend kitchen
\
in the early childhood classroom i have had several kids ask
```

```
me can we try cooking with real food i will take their idea \
and create common core cooking lessons where we learn importa
nt math and writing concepts while cooking delicious healthy
\
food for snack time my students will have a grounded apprecia
tion for the work that went into making the food and knowledg
e \
of where the ingredients came from as well as how it is healt
hy for their bodies this project would expand our learning of
 \
nutrition and agricultural cooking recipes by having us peel
our own apples to make homemade applesauce make our own bread
 \
and mix up healthy plants from our classroom garden in the sp
ring we will also create our own cookbooks to be printed and
\
shared with families students will gain math and literature s
kills as well as a life long enjoyment for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

ss
```

Out[28]:

```
{'neg': 0.01, 'neu': 0.745, 'pos': 0.245, 'com
pound': 0.9975}
```

In [29]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
neg=[];pos=[];neu=[]; compound = []

for i in tqdm(range(len(project_data['preprocessed_essays']))
):
    sentiment_scores = analyzer.polarity_scores(project_data[
'preprocessed_essays'][i])
    neg.append(sentiment_scores['neg'])
    pos.append(sentiment_scores['pos'])
    neu.append(sentiment_scores['neu'])
    compound.append(sentiment_scores['compound'])
```

```
100%|██████████| 109248/109248 [02:21<00:00, 7
74.21it/s]
```

In [30]:

```
#new columns indicating the sentiment score of each project e
ssay
project_data['neg'] = neg
project_data['neu'] = neu
project_data['pos'] = pos
project_data['compound'] = compound
```

In [31]:

```
project_data.head()
```

Out[31]:

	Unnamed: 0	id	teacher_id	project_submitted_date
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	2016-12-05 13:43:57
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	2016-10-25 09:2
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	2016-08-31 12:03:56

3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	2016-10-06 21:1
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	2016-07-11 01:10:09

1.4 Preprocessing of $project_t it \leq$

In [32]:

```
# similarly you can preprocess the titles also
preprocessed_titles = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['project_title'].values):
    sent = decontracted(sentence)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopw
ords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:01<00:00, 5
5961.83it/s]
```

In [33]:

```
#creating a new column with the preprocessed titles,useful fo
r analysis
project_data['preprocessed_titles'] = preprocessed_titles
```

In [34]:

```
title_word_count=[]
for i in range(len(project_data['preprocessed_titles'])):
    title_word_count.append(len(project_data['preprocessed_ti
tles'][i].split()))
```

In [35]:

```
project_data['title_word_count'] = title_word_count
```

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

In [36]:

```
from sklearn.model_selection import train_test_split
#How to split whole dataset into Train,CV and test
#https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html#sklearn.model_selection.train_test_split
project_data_train, project_data_test, y_train, y_test = train_test_split(project_data, project_data['project_is_approved'], test_size=0.33, stratify = project_data['project_is_approved'])
project_data_train, project_data_cv, y_train, y_cv = train_test_split(project_data_train, y_train, test_size=0.33, stratify=y_train)
```

In [37]:

```
print("Split ratio")
print('-'*50)
print('Train dataset:',len(project_data_train)/len(project_data)*100,'%\n','size:',len(project_data_train))
print('Cross validation dataset:',len(project_data_cv)/len(project_data)*100,'%\n','size:',len(project_data_cv))
print('Test dataset:',len(project_data_test)/len(project_data)*100,'%\n','size:',len(project_data_test))
```

```
Split ratio
----------------------------------------------
----
Train dataset: 44.889608963093146 %
 size: 49041
```

```
Cross validation dataset: 22.110244581136495 %
 size: 24155
Test dataset: 33.000146455770356 %
 size: 36052
```

In [38]:

```
#Features
project_data_train.drop(['project_is_approved'], axis=1, inpl
ace=True)
project_data_cv.drop(['project_is_approved'], axis=1, inplace
=True)
project_data_test.drop(['project_is_approved'], axis=1, inpla
ce=True)
```

1.5 Preparing data for models

In [39]:

```
project_data.columns
```

Out[39]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'proj
ect_submitted_datetime',
       'project_title', 'project_resource_summ
ary',
       'teacher_number_of_previously_posted_pr
ojects', 'project_is_approved',
       'clean_categories', 'clean_subcategorie
s', 'school_state',
       'teacher_prefix', 'project_grade_catego
ry', 'essay',
       'preprocessed_essays', 'essay_word_coun
t', 'neg', 'neu', 'pos',
       'compound', 'preprocessed_titles', 'tit
le_word_count'],
      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 : nu
merical
   - price : numerical
```

Make Data Model Ready: encoding numerical, categorical features

1.5.1 Vectorizing Categorical data

- 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_cat = CountVectorizer(vocabulary=list(sorted_cat_d
ict.keys()), lowercase=False, binary=True)
vectorizer_cat.fit(project_data_train['clean_categories'].val
ues) #fitting has to be on Train data

train_categories_one_hot = vectorizer_cat.transform(project_d
ata_train['clean_categories'].values)
cv_categories_one_hot = vectorizer_cat.transform(project_data
_cv['clean_categories'].values)
test_categories_one_hot = vectorizer_cat.transform(project_da
ta_test['clean_categories'].values)


print(vectorizer_cat.get_feature_names())
print("Shape of training data matrix after one hot encoding "
,train_categories_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_categories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'M
usic_Arts', 'AppliedLearning', 'SpecialNeeds',
 'Health_Sports', 'Math_Science', 'Literacy_La
nguage']
Shape of training data matrix after one hot en
coding  (49041, 9)
Shape of cross validation data matrix after on
e hot encoding  (24155, 9)
Shape of test data matrix after one hot encodi
ng  (36052, 9)
```

In [41]:

```
# we use count vectorizer to convert the values into one
vectorizer_subcat = CountVectorizer(vocabulary=list(sorted_su
b_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_subcat.fit(project_data_train['clean_subcategories'
].values)


train_subcategories_one_hot = vectorizer_subcat.transform(pro
ject_data_train['clean_subcategories'].values)
cv_subcategories_one_hot = vectorizer_subcat.transform(projec
t_data_cv['clean_subcategories'].values)
test_subcategories_one_hot = vectorizer_subcat.transform(proj
ect_data_test['clean_subcategories'].values)


print(vectorizer_subcat.get_feature_names())


print("Shape of train data matrix after one hot encoding ",tr
ain_subcategories_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_subcategories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_subcategories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLi
teracy', 'ParentInvolvement', 'Extracurricular
', 'Civics_Government', 'ForeignLanguages', 'N
utritionEducation', 'Warmth', 'Care_Hunger', '
SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'Other', 'College_Car
eerPrep', 'Music', 'History_Geography', 'Healt
h_LifeScience', 'EarlyDevelopment', 'ESL', 'Gy
m_Fitness', 'EnvironmentalScience', 'VisualArt
s', 'Health_Wellness', 'AppliedSciences', 'Spe
cialNeeds', 'Literature_Writing', 'Mathematics
', 'Literacy']
Shape of train data matrix after one hot encod
ing  (49041, 30)
Shape of cross validation data matrix after on
e hot encoding  (24155, 30)
Shape of test data matrix after one hot encodi
ng  (36052, 30)
```

In [42]:

```
## we use count vectorizer to convert the values into one hot
 encoded features

vectorizer_school_state = CountVectorizer()
vectorizer_school_state.fit(project_data_train['school_state']
.values)




print(vectorizer_school_state.get_feature_names())

train_school_state_category_one_hot = vectorizer_school_state
.transform(project_data_train['school_state'].values)
```

```
cv_school_state_category_one_hot = vectorizer_school_state.tr
ansform(project_data_cv['school_state'].values)
test_school_state_category_one_hot = vectorizer_school_state.
transform(project_data_test['school_state'].values)



print("Shape of train data matrix after one hot encoding ",tr
ain_school_state_category_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_school_state_category_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_school_state_category_one_hot.shape)
```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc
', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', '
in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi',
 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh
', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'or', '
pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va',
 'vt', 'wa', 'wi', 'wv', 'wy']
Shape of train data matrix after one hot encod
ing  (49041, 51)
Shape of cross validation data matrix after on
e hot encoding  (24155, 51)
Shape of test data matrix after one hot encodi
ng  (36052, 51)
```

In [43]:

```
#This step is to intialize a vectorizer with vocab from train
 data
my_counter = Counter()
for project_grade in project_data_train['project_grade_catego
ry'].values:
    my_counter.update(project_grade.split())
```

In [44]:

```
project_grade_cat_dict = dict(my_counter)
sorted_project_grade_cat_dict = dict(sorted(project_grade_cat
_dict.items(), key=lambda kv: kv[1]))
```

In [45]:

```
## we use count vectorizer to convert the values into one hot
 encoded features

vectorizer_grade = CountVectorizer(vocabulary=list(sorted_pro
ject_grade_cat_dict.keys()), lowercase=False, binary=True)
vectorizer_grade.fit(project_data_train['project_grade_catego
ry'].values)




print(vectorizer_grade.get_feature_names())

train_project_grade_category_one_hot = vectorizer_grade.trans
form(project_data_train['project_grade_category'].values)
cv_project_grade_category_one_hot = vectorizer_grade.transfor
m(project_data_cv['project_grade_category'].values)
test_project_grade_category_one_hot = vectorizer_grade.transf
orm(project_data_test['project_grade_category'].values)


print("Shape of train data matrix after one hot encoding ",tr
ain_project_grade_category_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_project_grade_category_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_project_grade_category_one_hot.shape)
```

```
['Grades_9_12', 'Grades_6_8', 'Grades_3_5', 'G
rades_PreK_2']
Shape of train data matrix after one hot encod
ing  (49041, 4)
Shape of cross validation data matrix after on
```

```
e hot encoding  (24155, 4)
Shape of test data matrix after one hot encodi
ng  (36052, 4)
```

In [46]:

```
#https://stackoverflow.com/questions/39303912/tfidfvectorizer
-in-scikit-learn-valueerror-np-nan-is-an-invalid-document
#ValueError: np.nan is an invalid document, expected byte or
unicode string.
vectorizer_prefix = CountVectorizer()
vectorizer_prefix.fit(project_data_train['teacher_prefix'].va
lues.astype("U"))


print(vectorizer_prefix.get_feature_names())


train_teacher_prefix_categories_one_hot = vectorizer_prefix.t
ransform(project_data_train['teacher_prefix'].values.astype("
U"))
cv_teacher_prefix_categories_one_hot = vectorizer_prefix.tran
sform(project_data_cv['teacher_prefix'].values.astype("U"))
test_teacher_prefix_categories_one_hot = vectorizer_prefix.tr
ansform(project_data_test['teacher_prefix'].values.astype("U"
))

print("Shape of train data matrix after one hot encoding ",tr
ain_teacher_prefix_categories_one_hot.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_teacher_prefix_categories_one_hot.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_teacher_prefix_categories_one_hot.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of train data matrix after one hot encod
ing  (49041, 5)
Shape of cross validation data matrix after on
```

```
e hot encoding  (24155, 5)
Shape of test data matrix after one hot encodi
ng  (36052, 5)
```

Make Data Model Ready: encoding essay, and project_title

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [47]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer_bow_essay = CountVectorizer(min_df=10)
vectorizer_bow_essay.fit(project_data_train['preprocessed_essays'].values)  #Fitting has to be on Train data


train_essay_bow = vectorizer_bow_essay.transform(project_data_train['essay'].values)
cv_essay_bow = vectorizer_bow_essay.transform(project_data_cv['essay'].values)
test_essay_bow = vectorizer_bow_essay.transform(project_data_test['essay'].values)


print("Shape of train data matrix after one hot encoding ",train_essay_bow.shape)
print("Shape of cross validation data matrix after one hot encoding ",cv_essay_bow.shape)
print("Shape of test data matrix after one hot encoding ",test_essay_bow.shape)
```

```
Shape of train data matrix after one hot encod
ing  (49041, 12107)
Shape of cross validation data matrix after on
e hot encoding  (24155, 12107)
Shape of test data matrix after one hot encodi
ng  (36052, 12107)
```

In [48]:

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
vectorizer_bow_title = CountVectorizer(min_df=10)
vectorizer_bow_title.fit_transform(project_data_train['prepro
cessed_titles'].values)    #Fitting has to be on Train data


train_title_bow = vectorizer_bow_title.transform(project_data
_train['preprocessed_titles'].values)
cv_title_bow = vectorizer_bow_title.transform(project_data_cv
['preprocessed_titles'].values)
test_title_bow = vectorizer_bow_title.transform(project_data_
test['preprocessed_titles'].values)



print("Shape of train data matrix after one hot encoding ",tr
ain_title_bow.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_title_bow.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_title_bow.shape)
```

```
Shape of train data matrix after one hot encod
ing  (49041, 2071)
Shape of cross validation data matrix after on
e hot encoding  (24155, 2071)
Shape of test data matrix after one hot encodi
ng  (36052, 2071)
```

1.5.2.2 TFIDF vectorizer

In [49]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essay = TfidfVectorizer(min_df=10)
vectorizer_tfidf_essay.fit(project_data_train['preprocessed_e
ssays'])      #Fitting has to be on Train data

train_essay_tfidf = vectorizer_tfidf_essay.transform(project_
data_train['preprocessed_essays'].values)
cv_essay_tfidf = vectorizer_tfidf_essay.transform(project_dat
a_cv['preprocessed_essays'].values)
test_essay_tfidf = vectorizer_tfidf_essay.transform(project_d
ata_test['preprocessed_essays'].values)

print("Shape of train data matrix after one hot encoding ",tr
ain_essay_tfidf.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_essay_tfidf.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_essay_tfidf.shape)
```

```
Shape of train data matrix after one hot encod
ing  (49041, 12107)
Shape of cross validation data matrix after on
e hot encoding  (24155, 12107)
Shape of test data matrix after one hot encodi
ng  (36052, 12107)
```

In [50]:

```
vectorizer_tfidf_title = TfidfVectorizer( min_df=10)
vectorizer_tfidf_title.fit(project_data_train['preprocessed_t
itles'])      #Fitting has to be on Train data
```

```
train_title_tfidf = vectorizer_tfidf_title.transform(project_
data_train['preprocessed_titles'].values)
cv_title_tfidf = vectorizer_tfidf_title.transform(project_dat
a_cv['preprocessed_titles'].values)
test_title_tfidf = vectorizer_tfidf_title.transform(project_d
ata_test['preprocessed_titles'].values)

print("Shape of train data matrix after one hot encoding ",tr
ain_title_tfidf.shape)
print("Shape of cross validation data matrix after one hot en
coding ",cv_title_tfidf.shape)
print("Shape of test data matrix after one hot encoding ",tes
t_title_tfidf.shape)
```

```
Shape of train data matrix after one hot encod
ing  (49041, 2071)
Shape of cross validation data matrix after on
e hot encoding  (24155, 2071)
Shape of test data matrix after one hot encodi
ng  (36052, 2071)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [51]:

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

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

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

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

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

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

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vec
tors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(word
s)*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.je
ssicayung.com/how-to-use-pickle-to-save-and-load-variables-in
-python/

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


'''
```

Out[51]:

```
'\n# Reading glove vectors in python: https://
stackoverflow.com/a/38230349/4084039\ndef load
GloveModel(gloveFile):\n    print ("Loading Gl
ove Model")\n    f = open(gloveFile,\'r\', enc
oding="utf8")\n    model = {}\n    for line in
 tqdm(f):\n        splitLine = line.split()\n
       word = splitLine[0]\n        embedding
= np.array([float(val) for val in splitLine[1:
]])\n        model[word] = embedding\n    prin
t ("Done.",len(model)," words loaded!")\n    r
eturn model\nmodel = loadGloveModel(\'glove.42
B.300d.txt\')\n\n# ===========================
=\nOutput:\n    \nLoading Glove Model\n1917495
it [06:32, 4879.69it/s]\nDone. 1917495  words
loaded!\n\n# ============================\n\nw
ords = []\nfor i in preproced_texts:\n    word
s.extend(i.split(\' \'))\n\nfor i in preproced
_titles:\n    words.extend(i.split(\' \'))\npr
int("all the words in the coupus", len(words))
\nwords = set(words)\nprint("the unique words
in the coupus", len(words))\n\ninter_words = s
et(model.keys()).intersection(words)\nprint("T
he number of words that are present in both gl
ove vectors and our coupus",       len(inter_w
ords),"(",np.round(len(inter_words)/len(words)
*100,3),"%)")\n\nwords_courpus = {}\nwords_glo
ve = set(model.keys())\nfor i in words:\n    i
f i in words_glove:\n        words_courpus[i]
= model[i]\nprint("word 2 vec length", len(wor
ds_courpus))\n\n\n# stronging variables into p
ickle files python: http://www.jessicayung.com
/how-to-use-pickle-to-save-and-load-variables-
in-python/\n\nimport pickle\nwith open(\'glove
_vectors\', \'wb\') as f:\n    pickle.dump(wor
ds_courpus, f)\n\n\n'
```

In [52]:

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

In [53]:

```
# average Word2Vec
# compute average word2vec for each review.
train_avg_w2v_essays = []; # the avg-w2v for each sentence/re
view is stored in this list
for sentence in tqdm(project_data_train['preprocessed_essays'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/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
    train_avg_w2v_essays.append(vector)

print(len(train_avg_w2v_essays))
print(len(train_avg_w2v_essays[0]))
```

```
100%|██████████| 49041/49041 [00:08<00:00, 583
1.10it/s]
```

```
49041
300
```

In [54]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_avg_w2v_essays = []; # the avg-w2v for each sentence/revie
w is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_essays']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/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
    cv_avg_w2v_essays.append(vector)
```

```
print(len(cv_avg_w2v_essays))
print(len(cv_avg_w2v_essays[0]))
```

```
100%|██████████| 24155/24155 [00:04<00:00, 586
8.97it/s]
```

```
24155
300
```

In [55]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_essays = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sentence in tqdm(project_data_test['preprocessed_essays']
): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/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
    test_avg_w2v_essays.append(vector)

print(len(test_avg_w2v_essays))
print(len(test_avg_w2v_essays[0]))
```

```
100%|██████████| 36052/36052 [00:06<00:00, 586
4.42it/s]
```

```
36052
300
```

In [56]:

```
# average Word2Vec
# compute average word2vec for each review.
train_avg_w2v_titles = []; # the avg-w2v for each sentence/re
view is stored in this list
for sentence in tqdm(project_data_train['preprocessed_titles'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/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
    train_avg_w2v_titles.append(vector)

print(len(train_avg_w2v_titles))
print(len(train_avg_w2v_titles[0]))
```

```
100%|██████████| 49041/49041 [00:00<00:00, 102
451.33it/s]
```

```
49041
300
```

In [57]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_avg_w2v_titles = []; # the avg-w2v for each sentence/revie
w is stored in this list
```

```
for sentence in tqdm(project_data_cv['preprocessed_titles']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/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
    cv_avg_w2v_titles.append(vector)

print(len(cv_avg_w2v_titles))
print(len(cv_avg_w2v_titles[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 100
168.69it/s]
```

```
24155
300
```

In [58]:

```
# average Word2Vec
# compute average word2vec for each review.
test_avg_w2v_titles = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sentence in tqdm(project_data_test['preprocessed_titles']
): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    cnt_words =0; # num of words with a valid vector in the s
entence/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
    test_avg_w2v_titles.append(vector)

print(len(test_avg_w2v_titles))
print(len(test_avg_w2v_titles[0]))
```

```
100%|██████████| 36052/36052 [00:00<00:00, 100
350.23it/s]
```

```
36052
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [59]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocessed_essays'].val
ues)
# we are converting a dictionary with word as a key, and the
idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [60]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_essays = []; # the avg-w2v for each sentence/
```

```
review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_essays'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_essays.append(vector)

print(len(train_tfidf_w2v_essays))
print(len(train_tfidf_w2v_essays[0]))
```

```
100%|██████████| 49041/49041 [01:02<00:00, 781
.95it/s]
```

```
49041
300
```

In [61]:

```
# average Word2Vec
# compute average word2vec for each review.
```

```
cv_tfidf_w2v_essays = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_essays']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    cv_tfidf_w2v_essays.append(vector)

print(len(cv_tfidf_w2v_essays))
print(len(cv_tfidf_w2v_essays[0]))
```

```
100%|██████████| 24155/24155 [00:30<00:00, 780
.53it/s]
```

```
24155
300
```

In [62]:

```
# average Word2Vec
```

```python
# compute average word2vec for each review.
test_tfidf_w2v_essays = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(project_data_test['preprocessed_essays']): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_essays.append(vector)

print(len(test_tfidf_w2v_essays))
print(len(test_tfidf_w2v_essays[0]))
```

```
100%|██████████| 36052/36052 [00:46<00:00, 780.80it/s]
```

```
36052
300
```

In [63]:

```
# Similarly you can vectorize for title also
tfidf_model = TfidfVectorizer()
tfidf_model.fit(project_data_train['preprocessed_titles'])
# we are converting a dictionary with word as a key, and the
idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(t
fidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [64]:

```
# average Word2Vec
# compute average word2vec for each review.
train_tfidf_w2v_titles = []; # the avg-w2v for each sentence/
review is stored in this list
for sentence in tqdm(project_data_train['preprocessed_titles'
]): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    train_tfidf_w2v_titles.append(vector)
```

```
print(len(train_tfidf_w2v_titles))
print(len(train_tfidf_w2v_titles[0]))
```

```
100%|██████████| 49041/49041 [00:01<00:00, 445
74.12it/s]
```

```
49041
300
```

In [65]:

```
# average Word2Vec
# compute average word2vec for each review.
cv_tfidf_w2v_titles = []; # the avg-w2v for each sentence/rev
iew is stored in this list
for sentence in tqdm(project_data_cv['preprocessed_titles']):
 # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    cv_tfidf_w2v_titles.append(vector)

print(len(cv_tfidf_w2v_titles))
print(len(cv_tfidf_w2v_titles[0]))
```

```
100%|██████████| 24155/24155 [00:00<00:00, 449
06.44it/s]
```

```
24155
300
```

In [66]:

```
# average Word2Vec
# compute average word2vec for each review.
test_tfidf_w2v_titles = []; # the avg-w2v for each sentence/r
eview is stored in this list
for sentence in tqdm(project_data_test['preprocessed_titles']
): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero leng
th
    tf_idf_weight =0; # num of words with a valid vector in t
he 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 w
ord
            # here we are multiplying idf value(dictionary[wo
rd]) and the tf value((sentence.count(word)/len(sentence.spli
t())))
            tf_idf = dictionary[word]*(sentence.count(word)/l
en(sentence.split())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weig
hted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    test_tfidf_w2v_titles.append(vector)

print(len(test_tfidf_w2v_titles))
print(len(test_tfidf_w2v_titles[0]))
```

```
100%|██████████| 36052/36052 [00:00<00:00, 466
84.83it/s]
```

```
36052
300
```

1.5.3 Vectorizing Numerical features

In [67]:

```
price_data = resource_data.groupby('id').agg({'price':'sum',
'quantity':'sum'}).reset_index()
```

In [68]:

```
project_data_train = pd.merge(project_data_train, price_data,
 on='id', how='left')
project_data_cv = pd.merge(project_data_cv, price_data, on='i
d', how='left')
project_data_test = pd.merge(project_data_test, price_data, o
n='id', how='left')
```

In [69]:

```
from sklearn.preprocessing import Normalizer
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array ins
tead:
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
```

```
normalizer = Normalizer()
normalizer.fit(project_data_train['price'].values.reshape(-1,1
))


price_normalized_train = normalizer.transform(project_data_tr
ain['price'].values.reshape(-1, 1))
price_normalized_cv = normalizer.transform(project_data_cv['p
rice'].values.reshape(-1, 1))
price_normalized_test = normalizer.transform(project_data_tes
t['price'].values.reshape(-1, 1))


print('After normalization')
print(price_normalized_train.shape)
print(price_normalized_cv.shape)
print(price_normalized_test.shape)
```

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

In [70]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['quantity'].values.reshape(-
1,1))


quantity_normalized_train = normalizer.transform(project_data
_train['quantity'].values.reshape(-1, 1))
quantity_normalized_cv = normalizer.transform(project_data_cv
['quantity'].values.reshape(-1, 1))
quantity_normalized_test = normalizer.transform(project_data_
test['quantity'].values.reshape(-1, 1))
```

```
print('After normalization')
print(quantity_normalized_train.shape)
print(quantity_normalized_cv.shape)
print(quantity_normalized_test.shape)
```

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

In [71]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['teacher_number_of_previous
ly_posted_projects'].values.reshape(-1,1))


previously_posted_projects_normalized_train = normalizer.tran
sform(project_data_train['teacher_number_of_previously_posted
_projects'].values.reshape(-1, 1))
previously_posted_projects_normalized_cv = normalizer.transfo
rm(project_data_cv['teacher_number_of_previously_posted_proje
cts'].values.reshape(-1, 1))
previously_posted_projects_normalized_test = normalizer.trans
form(project_data_test['teacher_number_of_previously_posted_p
rojects'].values.reshape(-1, 1))


print('After normalization')
print(previously_posted_projects_normalized_train.shape)
print(previously_posted_projects_normalized_cv.shape)
print(previously_posted_projects_normalized_test.shape)
```

```
After normalization
(49041, 1)
```

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

In [72]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['essay_word_count'].values.
reshape(-1,1))


essay_word_count_normalized_train = normalizer.transform(proj
ect_data_train['essay_word_count'].values.reshape(-1, 1))
essay_word_count_normalized_cv = normalizer.transform(project
_data_cv['essay_word_count'].values.reshape(-1, 1))
essay_word_count_normalized_test = normalizer.transform(proje
ct_data_test['essay_word_count'].values.reshape(-1, 1))


print('After normalization')
print(essay_word_count_normalized_train.shape)
print(essay_word_count_normalized_cv.shape)
print(essay_word_count_normalized_test.shape)
```

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

In [73]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['title_word_count'].values.
reshape(-1,1))


title_word_count_normalized_train = normalizer.transform(proj
ect_data_train['title_word_count'].values.reshape(-1, 1))
title_word_count_normalized_cv = normalizer.transform(project
_data_cv['title_word_count'].values.reshape(-1, 1))
title_word_count_normalized_test = normalizer.transform(proje
ct_data_test['title_word_count'].values.reshape(-1, 1))


print('After normalization')
print(title_word_count_normalized_train.shape)
print(title_word_count_normalized_cv.shape)
print(title_word_count_normalized_test.shape)
```

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

In [74]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['neg'].values.reshape(-1,1)
)


sent_neg_train = normalizer.transform(project_data_train['neg
'].values.reshape(-1, 1))
sent_neg_cv = normalizer.transform(project_data_cv['neg'].val
ues.reshape(-1, 1))
sent_neg_test = normalizer.transform(project_data_test['neg']
.values.reshape(-1, 1))


print('After normalization')
print(sent_neg_train.shape)
print(sent_neg_cv.shape)
print(sent_neg_test.shape)
```

```
After normalization
(49041, 1)
(24155, 1)
```

```
(36052, 1)
```

In [75]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['pos'].values.reshape(-1,1)
)


sent_pos_train = normalizer.transform(project_data_train['pos
'].values.reshape(-1, 1))
sent_pos_cv = normalizer.transform(project_data_cv['pos'].val
ues.reshape(-1, 1))
sent_pos_test = normalizer.transform(project_data_test['pos']
.values.reshape(-1, 1))


print('After normalization')
print(sent_pos_train.shape)
print(sent_pos_cv.shape)
print(sent_pos_test.shape)
```

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

In [76]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['neu'].values.reshape(-1,1)
)


sent_neu_train = normalizer.transform(project_data_train['neu
'].values.reshape(-1, 1))
sent_neu_cv = normalizer.transform(project_data_cv['neu'].val
ues.reshape(-1, 1))
```

```
sent_neu_test = normalizer.transform(project_data_test['neu']
.values.reshape(-1, 1))


print('After normalization')
print(sent_neu_train.shape)
print(sent_neu_cv.shape)
print(sent_neu_test.shape)
```

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

In [77]:

```
normalizer = Normalizer()
normalizer.fit(project_data_train['compound'].values.reshape(-
1,1))


sent_compound_train = normalizer.transform(project_data_train
['compound'].values.reshape(-1, 1))
sent_compound_cv = normalizer.transform(project_data_cv['comp
ound'].values.reshape(-1, 1))
sent_compound_test = normalizer.transform(project_data_test['
compound'].values.reshape(-1, 1))


print('After normalization')
print(sent_compound_train.shape)
print(sent_compound_cv.shape)
print(sent_compound_test.shape)
```

```
After normalization
(49041, 1)
(24155, 1)
```

```
(36052, 1)
```

Assignment 7: SVM

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

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

2. **The hyper 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 value
    - Find the best hyper paramter using k-fold cross validation or simple cross validation data
    - Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Representation of results**

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

- Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.
- Along with plotting ROC curve, you need to print the confusion matrix with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

4. **[Task-2] Apply 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 on TfidfVectorizer of essay text, choose the number of components (`n_components`) using elbow method** : 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

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.

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

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

Support Vector Machines

Set 1:Categorical, Numerical features + Project_title(BOW) + Preprocessed_essay (BOW)

In [79]:

```
# merge two sparse matrices: https://stackoverflow.com/a/1971
0648/4084039
from scipy.sparse import hstack

X_train = hstack((train_categories_one_hot, train_subcategori
es_one_hot, train_school_state_category_one_hot , train_proje
ct_grade_category_one_hot, train_teacher_prefix_categories_on
e_hot, price_normalized_train, quantity_normalized_train, pre
viously_posted_projects_normalized_train, title_word_count_no
rmalized_train, essay_word_count_normalized_train, sent_neg_t
rain,sent_pos_train,sent_neu_train,sent_compound_train , trai
n_title_bow, train_essay_bow)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_
one_hot, test_school_state_category_one_hot , test_project_gr
ade_category_one_hot, test_teacher_prefix_categories_one_hot,
 price_normalized_test, quantity_normalized_test, previously_
posted_projects_normalized_test, title_word_count_normalized_
test, essay_word_count_normalized_test, sent_neg_test,sent_po
s_test,sent_neu_test,sent_compound_test , test_title_bow, tes
t_essay_bow )).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_ho
t, cv_school_state_category_one_hot , cv_project_grade_catego
ry_one_hot, cv_teacher_prefix_categories_one_hot, price_norma
lized_cv, quantity_normalized_cv, previously_posted_projects_
normalized_cv, title_word_count_normalized_cv, essay_word_cou
nt_normalized_cv,sent_neg_cv,sent_pos_cv,sent_neu_cv,sent_com
pound_cv ,cv_title_bow, cv_essay_bow)).tocsr()
```

In [80]:

```
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(49041, 14286)
(24155, 14286)
(36052, 14286)
```

Train the model using the best penalty and best hyperparameter that will maximise AUC score using GridSearchCV

In [81]:

```
# https://medium.com/@erikgreenj/k-neighbors-classifier-with-
gridsearchcv-basics-3c445ddeb657

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
sv = SGDClassifier(max_iter=1000,tol=0.001,class_weight='bala
nced')
grid_params = {'alpha': [10**-4, 10**-3, 10**-2, 10**-1, 10**
0, 10**1, 10**2, 10**3, 10**4], 'penalty': ('l1','l2')}
gs = GridSearchCV(sv, grid_params, cv=10, scoring='roc_auc',)
gs_results = gs.fit(X_train, y_train)
print(gs_results.best_score_)
print(gs_results.best_estimator_)
print(gs_results.best_params_)
```

```
0.714785302278
SGDClassifier(alpha=0.01, average=False, class
_weight='balanced',
       early_stopping=False, epsilon=0.1, eta0
=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal',
 loss='hinge', max_iter=1000,
       n_iter=None, n_iter_no_change=5, n_jobs
=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle
=True, tol=0.001,
       validation_fraction=0.1, verbose=0, war
m_start=False)
```

```
{'alpha': 0.01, 'penalty': 'l2'}
```

In [82]:

```
print("Best score(AUC): ",gs_results.best_score_)
print('k value with best auc score: ',gs_results.best_params_
)
```

```
Best score(AUC):  0.714785302278
k value with best auc score:  {'alpha': 0.01,
'penalty': 'l2'}
```

In [83]:

```
best_pen = gs_results.best_params_['penalty']
best_alpha = gs_results.best_params_['alpha']
```

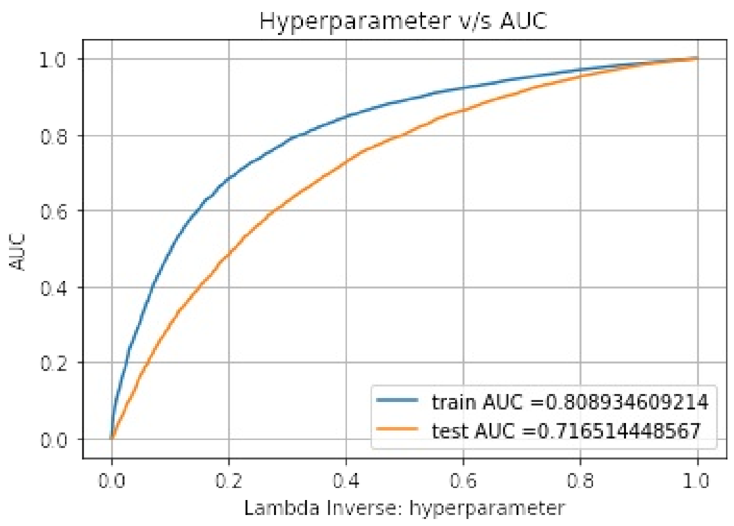
In [84]:

```
def pred_prob(clf, data):
    y_pred = []
    y_pred = clf.predict_proba(data)[:,1]
    return y_pred
```

In [85]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
etrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

sv = SGDClassifier(alpha = best_alpha ,max_iter=1000,tol=0.00
1,penalty= best_pen,class_weight='balanced')
calibrated_sv = CalibratedClassifierCV(base_estimator=sv, met
hod='isotonic')
calibrated_sv.fit(X_train,y_train)

y_train_pred = pred_prob(calibrated_sv,X_train)
y_test_pred = pred_prob(calibrated_sv,X_test)
```

```
train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.close
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("Lambda Inverse: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyperparameter v/s AUC")
plt.grid()
plt.show()
```



Confusion Matrix of Train and Test Data

In [86]:

```
# we are writing our own function for predict, with defined t
hreshold
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshold, fpr, tpr):
    t = threshold[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low a
nd tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)
), "for threshold", np.round(t,3))
    return t

def predict_with_best_t(proba, threshold):
    predictions = []
    for i in proba:
        if i>=threshold:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [87]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.55278316078
```

```
8 for threshold 0.831
Train confusion matrix
[[ 5659  1767]
 [11428 30187]]
```

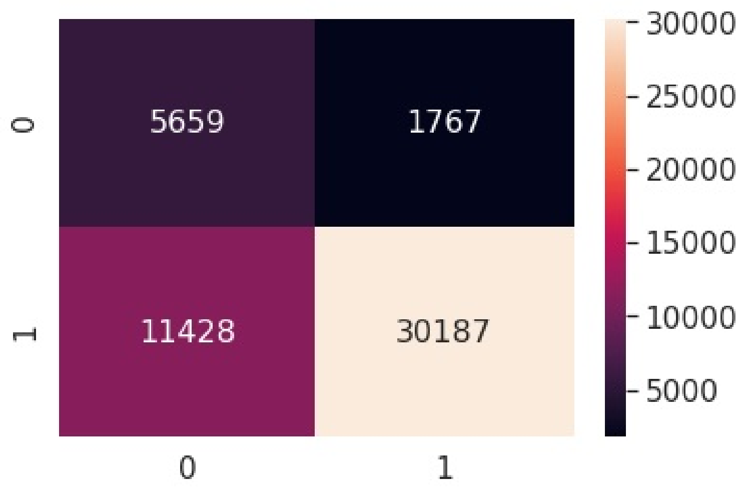
In [88]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_t
rain, predict_with_best_t(y_train_pred, best_t)), range(2),ra
nge(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={
"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[88]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a201ebef0>
```



In [89]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pre
d, best_t)))
```

```
Test confusion matrix
[[ 3403  2056]
 [ 9095 21498]]
```

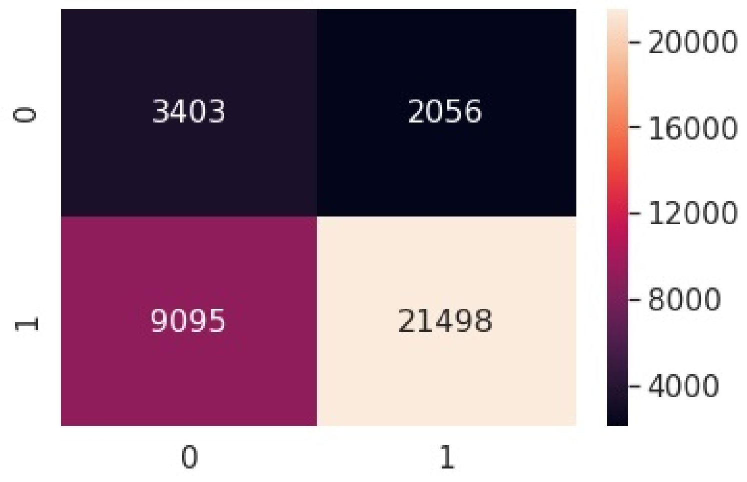
In [90]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_te
st, predict_with_best_t(y_test_pred, best_t)), range(2),range
(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"
size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[90]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a2f250d68>
```



Set 2 : Categorical, Numerical features + Project_title(TFIDF) + Preprocessed_essay (TFIDF)

In [91]:

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

X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_school_state_category_one_hot , train_project_grade_category_one_hot, train_teacher_prefix_categories_one_hot, price_normalized_train, quantity_normalized_train, previously_posted_projects_normalized_train, title_word_count_normalized_train, essay_word_count_normalized_train,sent_neg_train,sent_pos_train,sent_neu_train,sent_compound_train ,train_title_tfidf, train_essay_tfidf)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_school_state_category_one_hot , test_project_grade_category_one_hot, test_teacher_prefix_categories_one_hot, price_normalized_test, quantity_normalized_test, previously_posted_projects_normalized_test, title_word_count_normalized_test, essay_word_count_normalized_test, sent_neg_test,sent_pos_test,sent_neu_test,sent_compound_test,test_title_tfidf, test_essay_tfidf)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_hot, cv_school_state_category_one_hot , cv_project_grade_category_one_hot, cv_teacher_prefix_categories_one_hot, price_normalized_cv, quantity_normalized_cv, previously_posted_projects_normalized_cv, title_word_count_normalized_cv, essay_word_count_normalized_cv, sent_neg_cv,sent_pos_cv,sent_neu_cv,sent_compound_cv , cv_title_tfidf, cv_essay_tfidf)).tocsr()
```

In [92]:

```
print(X_train.shape)
print(X_test.shape)
print(X_cv.shape)
```

```
(49041, 14286)
(36052, 14286)
(24155, 14286)
```

Train the model using the best penalty and best hyperparameter that will maximise AUC score using GridSearchCV

In [93]:

```
# https://medium.com/@erikgreenj/k-neighbors-classifier-with-
gridsearchcv-basics-3c445ddeb657

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
sv = SGDClassifier(max_iter=1000,tol=0.001,class_weight='bala
nced')
grid_params = {'alpha': [10**-4, 10**-3, 10**-2, 10**-1, 10**
0, 10**1, 10**2, 10**3, 10**4], 'penalty': ('l1','l2')}
gs = GridSearchCV(sv, grid_params, cv=10, scoring='roc_auc',)
gs_results = gs.fit(X_train, y_train)
print(gs_results.best_score_)
print(gs_results.best_estimator_)
print(gs_results.best_params_)
```

```
0.703958211394
SGDClassifier(alpha=0.0001, average=False, cla
ss_weight='balanced',
       early_stopping=False, epsilon=0.1, eta0
=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal',
 loss='hinge', max_iter=1000,
       n_iter=None, n_iter_no_change=5, n_jobs
=None, penalty='l1',
       power_t=0.5, random_state=None, shuffle
=True, tol=0.001,
       validation_fraction=0.1, verbose=0, war
m_start=False)
{'alpha': 0.0001, 'penalty': 'l1'}
```

In [94]:

```
print("Best score (AUC): ",gs_results.best_score_)
print('k value with best auc score: ',gs_results.best_params_
)
```

```
Best score (AUC):  0.703958211394
k value with best auc score:  {'alpha': 0.0001
, 'penalty': 'l1'}
```

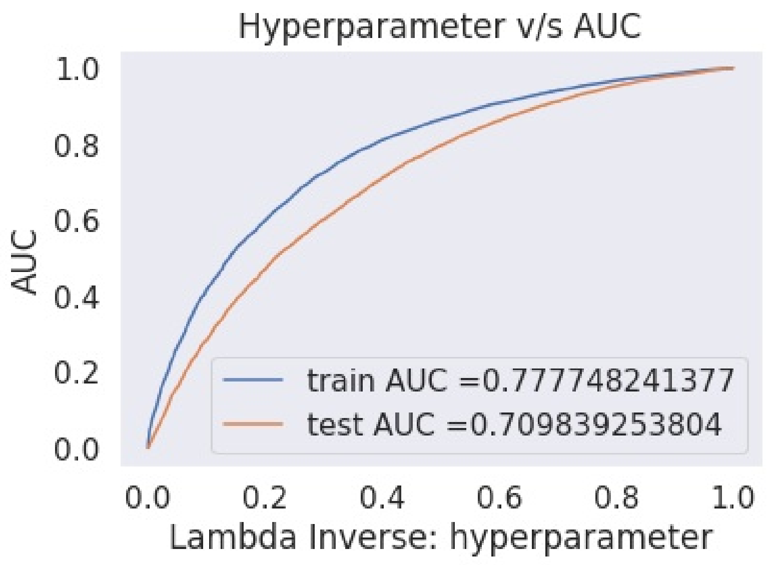
In [95]:

```
best_pen = gs_results.best_params_['penalty']
best_alpha = gs_results.best_params_['alpha']
```

In [96]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
etrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

sv = SGDClassifier(alpha = best_alpha ,max_iter=1000,tol=0.00
1,penalty= best_pen,class_weight='balanced')
calibrated_sv = CalibratedClassifierCV(base_estimator=sv, met
hod='isotonic')
calibrated_sv.fit(X_train,y_train)
```

```
y_train_pred = pred_prob(calibrated_sv,X_train)
y_test_pred = pred_prob(calibrated_sv,X_test)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.close
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("Lambda Inverse: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyperparameter v/s AUC")
plt.grid()
plt.show()
```



Confusion Matrix of Train and Test Data

In [97]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.50917945686
 for threshold 0.832
Train confusion matrix
[[ 5275  2151]
 [11785 29830]]
```

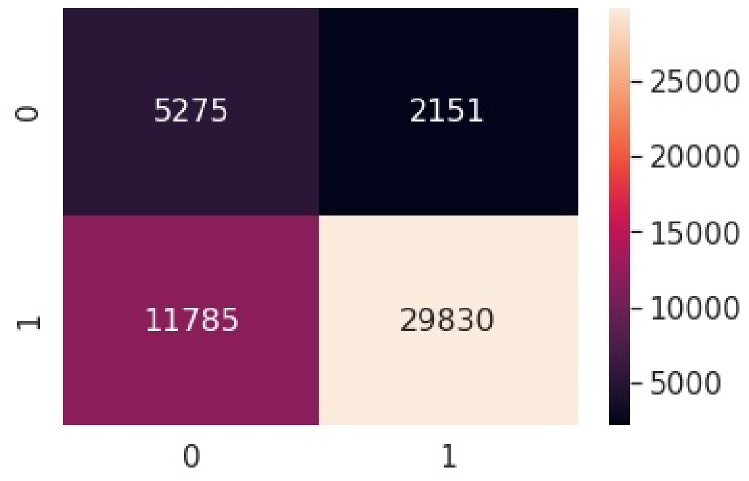
In [98]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_t
rain, predict_with_best_t(y_train_pred, best_t)), range(2),ra
nge(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={
"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[98]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a2f3ed550>
```



In [99]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pre
d, best_t)))
```

```
Test confusion matrix
[[ 3344  2115]
 [ 9261 21332]]
```

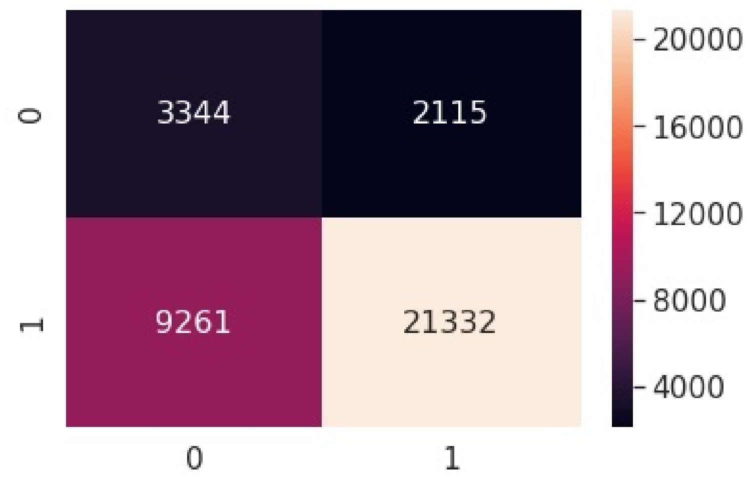
In [100]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_te
st, predict_with_best_t(y_test_pred, best_t)), range(2),range
(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"
size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[100]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a2f115780>
```



0
1
3344
2115
9261
21332
0
1
20000
16000
12000
8000
4000

Set 3: Categorical, Numerical features + Project_title(AVG W2V) + Preprocessed_essay (AVG W2V)

In [101]:

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

X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_school_state_category_one_hot , train_project_grade_category_one_hot, train_teacher_prefix_categories_one_hot, price_normalized_train, quantity_normalized_train, previously_posted_projects_normalized_train, title_word_count_normalized_train, essay_word_count_normalized_train,sent_neg_train,sent_pos_train,sent_neu_train,sent_compound_train , train_avg_w2v_titles, train_avg_w2v_essays)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_school_state_category_one_hot , test_project_grade_category_one_hot, test_teacher_prefix_categories_one_hot, price_normalized_test, quantity_normalized_test, previously_posted_projects_normalized_test, title_word_count_normalized_test, essay_word_count_normalized_test, sent_neg_test,sent_pos_test,sent_neu_test,sent_compound_test  , test_avg_w2v_titles, test_avg_w2v_essays)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_hot, cv_school_state_category_one_hot , cv_project_grade_category_one_hot, cv_teacher_prefix_categories_one_hot, price_normalized_cv, quantity_normalized_cv, previously_posted_projects_normalized_cv, title_word_count_normalized_cv, essay_word_count_normalized_cv, sent_neg_cv,sent_pos_cv,sent_neu_cv,sent_compound_cv  , cv_avg_w2v_titles, cv_avg_w2v_essays)).tocsr()
```

In [102]:

```
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(49041, 708)
(24155, 708)
(36052, 708)
```

Train the model using the best penalty and best hyperparameter that will maximise AUC score using GridSearchCV

In [103]:

```
# https://medium.com/@erikgreenj/k-neighbors-classifier-with-
gridsearchcv-basics-3c445ddeb657

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
sv = SGDClassifier(max_iter=1000,tol=0.001,class_weight='bala
nced')
grid_params = {'alpha': [10**-4, 10**-3, 10**-2, 10**-1, 10**
0, 10**1, 10**2, 10**3, 10**4], 'penalty': ('l1','l2')}
gs = GridSearchCV(sv, grid_params, cv=10, scoring='roc_auc',)
gs_results = gs.fit(X_train, y_train)
print(gs_results.best_score_)
print(gs_results.best_estimator_)
print(gs_results.best_params_)
```

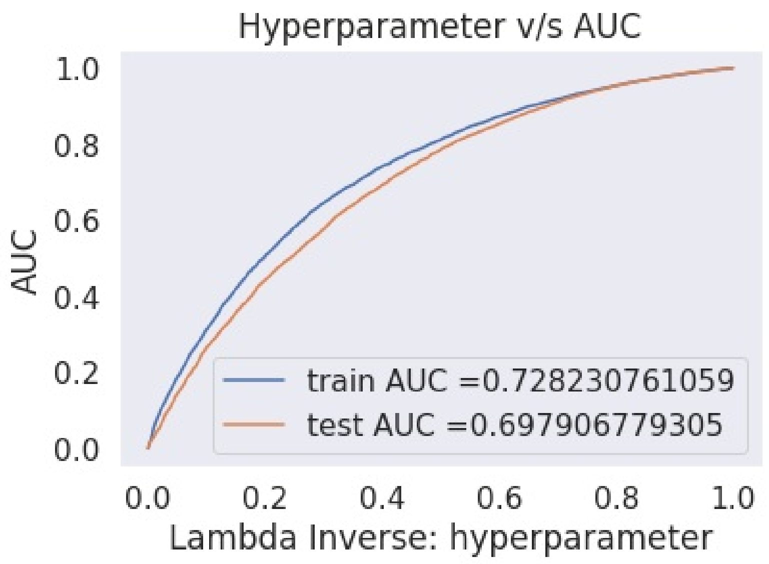
```
0.69024250143
SGDClassifier(alpha=0.0001, average=False, cla
ss_weight='balanced',
       early_stopping=False, epsilon=0.1, eta0
=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal',
 loss='hinge', max_iter=1000,
       n_iter=None, n_iter_no_change=5, n_jobs
=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle
=True, tol=0.001,
       validation_fraction=0.1, verbose=0, war
m_start=False)
{'alpha': 0.0001, 'penalty': 'l2'}
```

In [104]:

```
best_pen = gs_results.best_params_['penalty']
best_alpha = gs_results.best_params_['alpha']
```

In [105]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
etrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

sv = SGDClassifier(alpha = best_alpha ,max_iter=1000,tol=0.00
1,penalty= best_pen,class_weight='balanced')
calibrated_sv = CalibratedClassifierCV(base_estimator=sv, met
hod='isotonic')
calibrated_sv.fit(X_train,y_train)

y_train_pred = pred_prob(calibrated_sv,X_train)
y_test_pred = pred_prob(calibrated_sv,X_test)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.close
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("Lambda Inverse: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyperparameter v/s AUC")
plt.grid()
plt.show()
```



Confusion Matrix of Train and Test Data

In [106]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.45394782932
4 for threshold 0.843
Train confusion matrix
[[ 4937  2489]
 [13200 28415]]
```

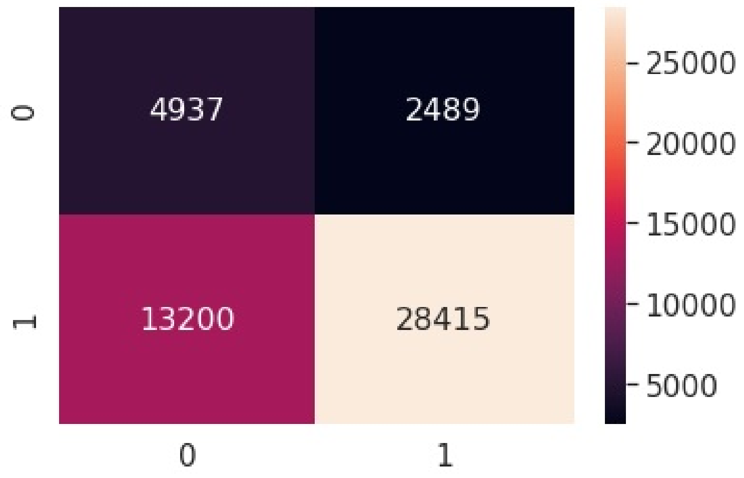
In [107]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_t
rain, predict_with_best_t(y_train_pred, best_t)), range(2),ra
nge(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={
"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[107]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a2f10f898>
```



In [108]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pre
d, best_t)))
```

```
Test confusion matrix
[[ 3337  2122]
 [ 9802 20791]]
```

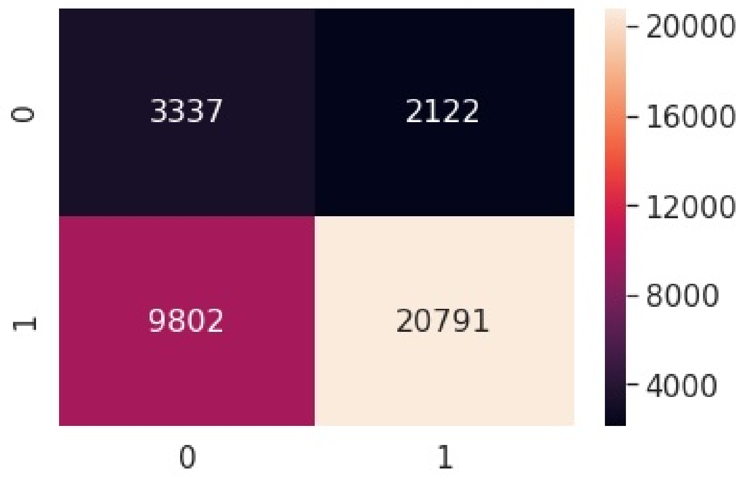
In [109]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_te
st, predict_with_best_t(y_test_pred, best_t)), range(2),range
(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"
size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a2f86a710>
```



Set 4 : Categorical, Numerical features + Project_title(TFIDF W2V) + Preprocessed_essay (TFIDF W2V)

In [110]:

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

X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_school_state_category_one_hot , train_project_grade_category_one_hot, train_teacher_prefix_categories_one_hot, price_normalized_train, quantity_normalized_train, previously_posted_projects_normalized_train, title_word_count_normalized_train, essay_word_count_normalized_train, sent_neg_train,sent_pos_train,sent_neu_train,sent_compound_train, train_tfidf_w2v_titles, train_tfidf_w2v_essays)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_school_state_category_one_hot , test_project_grade_category_one_hot, test_teacher_prefix_categories_one_hot, price_normalized_test, quantity_normalized_test, previously_posted_projects_normalized_test, title_word_count_normalized_test, essay_word_count_normalized_test, sent_neg_test,sent_pos_test,sent_neu_test,sent_compound_test , test_tfidf_w2v_titles, test_tfidf_w2v_essays)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_hot, cv_school_state_category_one_hot , cv_project_grade_category_one_hot, cv_teacher_prefix_categories_one_hot, price_normalized_cv, quantity_normalized_cv, previously_posted_projects_normalized_cv, title_word_count_normalized_cv, essay_word_count_normalized_cv, sent_neg_cv,sent_pos_cv,sent_neu_cv,sent_compound_cv  , cv_tfidf_w2v_titles, cv_tfidf_w2v_essays)).tocsr
```

```
()
```

In [111]:

```
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(49041, 708)
(24155, 708)
(36052, 708)
```

Train the model using the best penalty and best hyperparameter that will maximise AUC score using GridSearchCV

In [112]:

```
# https://medium.com/@erikgreenj/k-neighbors-classifier-with-
gridsearchcv-basics-3c445ddeb657

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
sv = SGDClassifier(max_iter=1000,tol=0.001,class_weight='bala
nced')
grid_params = {'alpha': [10**-4, 10**-3, 10**-2, 10**-1, 10**
0, 10**1, 10**2, 10**3, 10**4], 'penalty': ('l1','l2')}
gs = GridSearchCV(sv, grid_params, cv=10, scoring='roc_auc',)
gs_results = gs.fit(X_train, y_train)
print(gs_results.best_score_)
print(gs_results.best_estimator_)
print(gs_results.best_params_)
```

```
0.687859923073
SGDClassifier(alpha=0.001, average=False, clas
s_weight='balanced',
       early_stopping=False, epsilon=0.1, eta0
=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal',
 loss='hinge', max_iter=1000,
       n_iter=None, n_iter_no_change=5, n_jobs
=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle
=True, tol=0.001,
       validation_fraction=0.1, verbose=0, war
m_start=False)
{'alpha': 0.001, 'penalty': 'l2'}
```

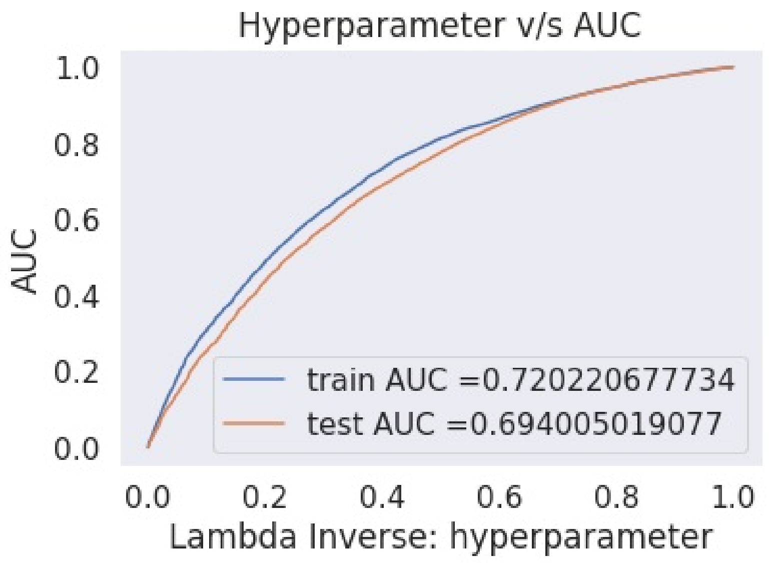
In [114]:

```
best_pen = gs_results.best_params_['penalty']
best_alpha = gs_results.best_params_['alpha']
```

In [115]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
etrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

sv = SGDClassifier(alpha = best_alpha ,max_iter=1000,tol=0.00
1,penalty= best_pen,class_weight='balanced')
calibrated_sv = CalibratedClassifierCV(base_estimator=sv, met
hod='isotonic')
calibrated_sv.fit(X_train,y_train)

y_train_pred = pred_prob(calibrated_sv,X_train)
y_test_pred = pred_prob(calibrated_sv,X_test)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)
```

```
plt.close
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("Lambda Inverse: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyperparameter v/s AUC")
plt.grid()
plt.show()
```



Confusion Matrix of Train and Test Data

In [116]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.44372901417
4 for threshold 0.837
Train confusion matrix
[[ 4649  2777]
 [12119 29496]]
```

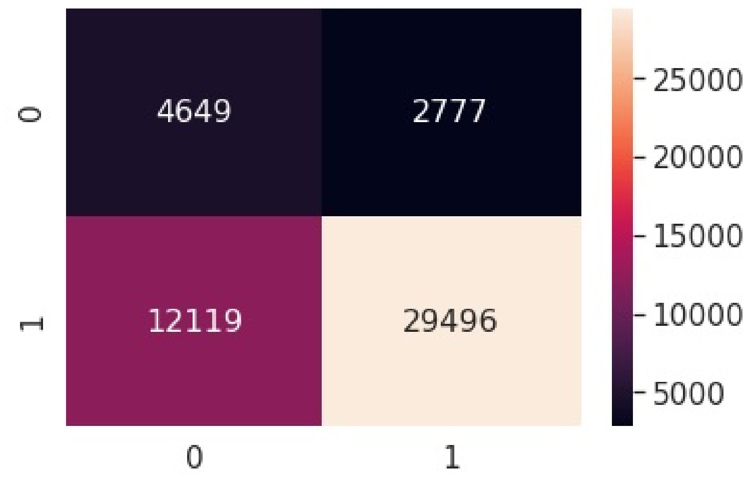
In [117]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_t
rain, predict_with_best_t(y_train_pred, best_t)), range(2),ra
nge(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={
"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[117]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a2f66b550>
```



In [118]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pre
d, best_t)))
```

```
Test confusion matrix
[[ 3223  2236]
 [ 9282 21311]]
```

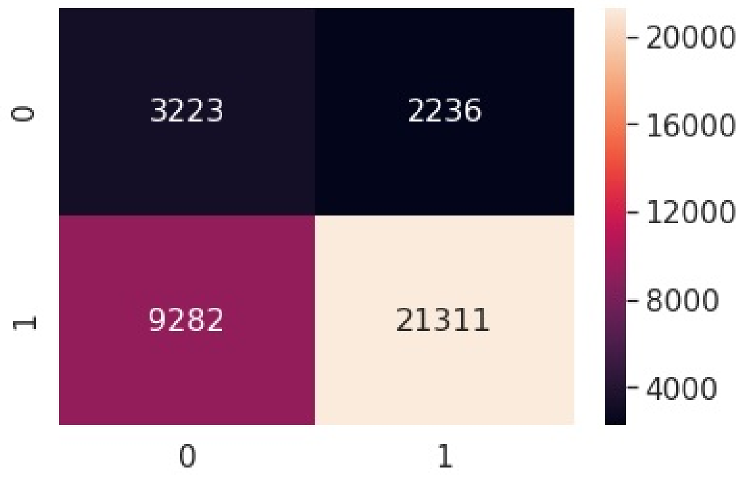
In [119]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_te
st, predict_with_best_t(y_test_pred, best_t)), range(2),range
(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"
size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[119]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a2f29b0b8>
```



0
1
3223
2236
9282
21311
0
1
20000
16000
12000
8000
4000

Set 5 : Apply TruncatedSVD on TfidfVectorizer of essay text, choose the number of components (`n_components`) using elbow method

In [120]:

```
# Create and run  TSVD
from sklearn.decomposition import TruncatedSVD
tsvd = TruncatedSVD(n_components=1500)
X_tfidf_tsvd = tsvd.fit(train_essay_tfidf)
```

Using Elbow method find out the best number of Components

In [121]:

```
# List of explained variances
tsvd_var_ratios = tsvd.explained_variance_ratio_
```

In [122]:

```
tsvd_var_ratios
```

Out[122]:

```
array([ 0.00393001,  0.01024491,  0.0087578 ,
...,  0.00013839,
        0.0001383 ,  0.00013799])
```

In [123]:

```
# citation link : https://chrisalbon.com/machine_learning/fea
ture_engineering/select_best_number_of_components_in_tsvd/
# Create Function Calculating Number Of Components Required T
o Pass Threshold
def select_n_components(var_ratio, goal_var: float) -> int:
    # Set initial variance explained so far
    total_variance = 0.0

    # Set initial number of features
    n_components = 0

    # For the explained variance of each feature:
    for explained_variance in var_ratio:

        # Add the explained variance to the total
        total_variance += explained_variance

        # Add one to the number of components
        n_components += 1

        # If we reach our goal level of explained variance
        if total_variance >= goal_var:
            # End the loop
            break

    # Return the number of components
    return n_components
```

In [124]:

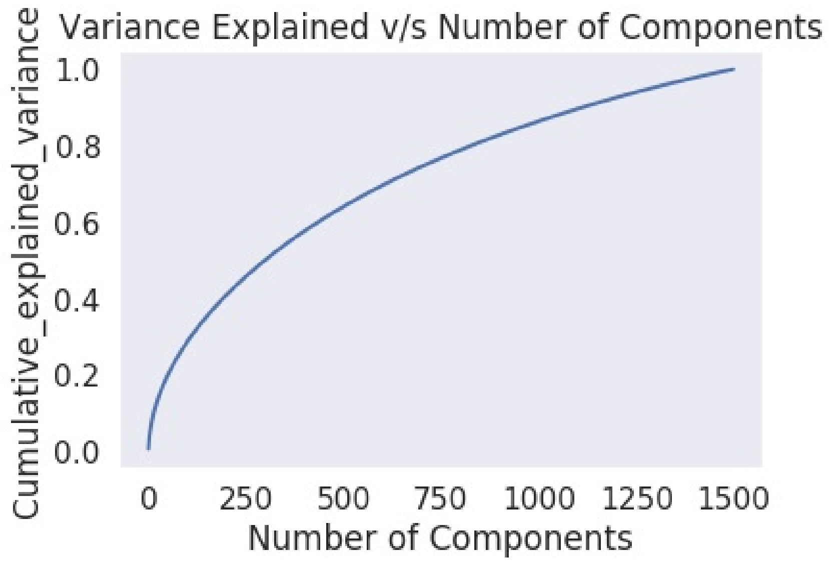
```
# Run function
select_n_components(tsvd_var_ratios, 0.95)
```

Out[124]:

```
1500
```

In [125]:

```
percentage_var_explained = tsvd.explained_variance_ / np.sum(
tsvd.explained_variance_);

cum_var_explained = np.cumsum(percentage_var_explained)

# Plot the PCA spectrum
plt.figure(1, figsize=(6, 4))

plt.clf()
plt.plot(cum_var_explained, linewidth=2)
plt.axis('tight')
plt.grid()
plt.xlabel("Number of Components")
plt.ylabel("Cumulative_explained_variance")
plt.title("Variance Explained v/s Number of Components")
plt.show()
```



So from the plot, it is evident that if we take around top 1500 features, we are able to preserve most of the variance

In [126]:

```
tsvd_final = TruncatedSVD(n_components= 1500, n_iter=7, rando
m_state=42)
tsvd_final.fit(train_essay_tfidf)
tsvd_tfidf_svd_train  =  tsvd_final.transform(train_essay_tfi
df)
tsvd_tfidf_svd_cv  =  tsvd_final.transform(cv_essay_tfidf)
tsvd_tfidf_svd_test  =  tsvd_final.transform(test_essay_tfidf
)
```

In [127]:

```
print(tsvd_tfidf_svd_train.shape)
print(tsvd_tfidf_svd_cv.shape)
print(tsvd_tfidf_svd_test.shape)
```

```
(49041, 1500)
(24155, 1500)
(36052, 1500)
```

Set 5: Categorical + Numerical features + TruncatedSVD vectorizer applied on tfidf vectorized essays

In [128]:

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

X_train = hstack((train_categories_one_hot, train_subcategories_one_hot, train_school_state_category_one_hot , train_project_grade_category_one_hot, train_teacher_prefix_categories_one_hot, price_normalized_train, quantity_normalized_train, previously_posted_projects_normalized_train, title_word_count_normalized_train, essay_word_count_normalized_train, sent_neg_train,sent_pos_train,sent_neu_train,sent_compound_train, tsvd_tfidf_svd_train)).tocsr()
X_test = hstack((test_categories_one_hot, test_subcategories_one_hot, test_school_state_category_one_hot , test_project_grade_category_one_hot, test_teacher_prefix_categories_one_hot, price_normalized_test, quantity_normalized_test, previously_posted_projects_normalized_test, title_word_count_normalized_test, essay_word_count_normalized_test, sent_neg_test,sent_pos_test,sent_neu_test,sent_compound_test , tsvd_tfidf_svd_test)).tocsr()
X_cv = hstack((cv_categories_one_hot, cv_subcategories_one_hot, cv_school_state_category_one_hot , cv_project_grade_category_one_hot, cv_teacher_prefix_categories_one_hot, price_normalized_cv, quantity_normalized_cv, previously_posted_projects_normalized_cv, title_word_count_normalized_cv, essay_word_count_normalized_cv, sent_neg_cv,sent_pos_cv,sent_neu_cv,sent_compound_cv  , tsvd_tfidf_svd_cv)).tocsr()
```

In [129]:

```
print(X_train.shape)
print(X_cv.shape)
print(X_test.shape)
```

```
(49041, 1608)
(24155, 1608)
(36052, 1608)
```

Train the model using the best penalty and best hyperparameter that will maximise AUC score using GridSearchCV

In [130]:

```
# https://medium.com/@erikgreenj/k-neighbors-classifier-with-
gridsearchcv-basics-3c445ddeb657

from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import SGDClassifier
sv = SGDClassifier(max_iter=1000,tol=0.001,class_weight='bala
nced')
grid_params = {'alpha': [10**-4, 10**-3, 10**-2, 10**-1, 10**
0, 10**1, 10**2, 10**3, 10**4], 'penalty': ('l1','l2')}
gs = GridSearchCV(sv, grid_params, cv=10, scoring='roc_auc',)
gs_results = gs.fit(X_train, y_train)
print(gs_results.best_score_)
print(gs_results.best_estimator_)
print(gs_results.best_params_)
```

```
0.708427551216
SGDClassifier(alpha=0.0001, average=False, cla
ss_weight='balanced',
       early_stopping=False, epsilon=0.1, eta0
=0.0, fit_intercept=True,
       l1_ratio=0.15, learning_rate='optimal',
 loss='hinge', max_iter=1000,
       n_iter=None, n_iter_no_change=5, n_jobs
=None, penalty='l2',
       power_t=0.5, random_state=None, shuffle
=True, tol=0.001,
       validation_fraction=0.1, verbose=0, war
m_start=False)
{'alpha': 0.0001, 'penalty': 'l2'}
```

In [131]:

```
print("Best score (AUC): ",gs_results.best_score_)
print('k value with best auc score: ',gs_results.best_params_
)
```

```
Best score (AUC):  0.708427551216
k value with best auc score:  {'alpha': 0.0001
, 'penalty': 'l2'}
```

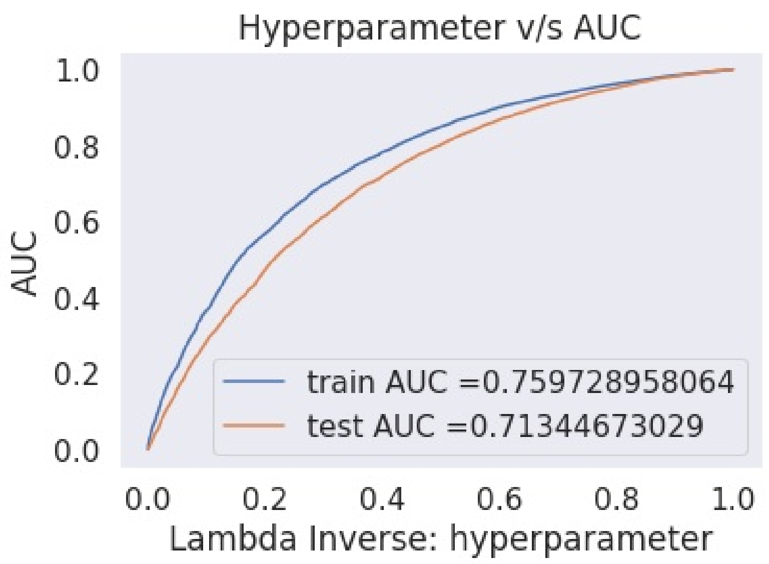
In [132]:

```
best_pen = gs_results.best_params_['penalty']
best_alpha = gs_results.best_params_['alpha']
```

In [135]:

```
# https://scikit-learn.org/stable/modules/generated/sklearn.m
etrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc
from sklearn.calibration import CalibratedClassifierCV

sv = SGDClassifier(alpha = best_alpha ,max_iter=1000,tol=0.00
1,penalty= best_pen,class_weight='balanced')
calibrated_sv = CalibratedClassifierCV(base_estimator=sv, met
hod='isotonic')
calibrated_sv.fit(X_train,y_train)
```

```
y_train_pred = pred_prob(calibrated_sv,X_train)
y_test_pred = pred_prob(calibrated_sv,X_test)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_tr
ain_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_
pred)

plt.close
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(tr
ain_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_
fpr, test_tpr)))
plt.legend()
plt.xlabel("Lambda Inverse: hyperparameter")
plt.ylabel("AUC")
plt.title("Hyperparameter v/s AUC")
plt.grid()
plt.show()
```



Confusion Matrix of Train and Test Data

In [136]:

```
#our objective here is to make auc the maximum
#so we find  the best threshold that will give the least fpr
best_t = find_best_threshold(tr_thresholds, train_fpr, train_
tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_p
red, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.48863507420
4 for threshold 0.841
Train confusion matrix
[[ 5211  2215]
 [12637 28978]]
```

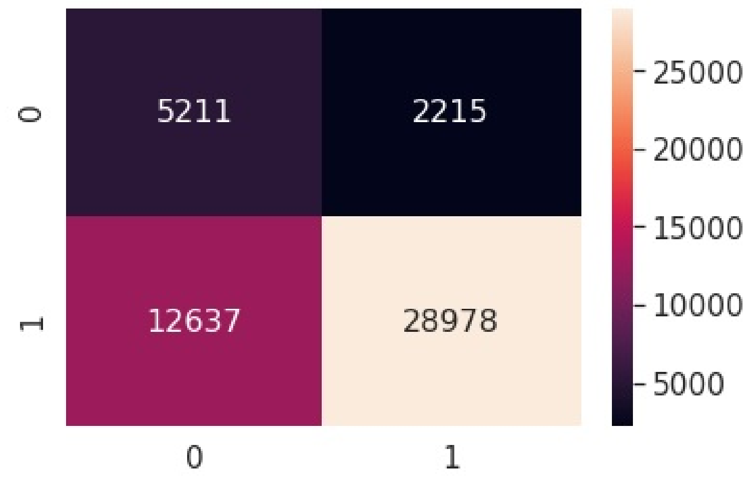
In [137]:

```
#plotting confusion matrix using seaborn's heatmap
# https://stackoverflow.com/questions/35572000/how-can-i-plot
-a-confusion-matrix

print("Train data confusion matrix")

confusion_matrix_df_train = pd.DataFrame(confusion_matrix(y_t
rain, predict_with_best_t(y_train_pred, best_t)), range(2),ra
nge(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_train, annot=True,annot_kws={
"size": 16}, fmt='g')
```

```
Train data confusion matrix
```

Out[137]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a2f2216a0>
```



In [138]:

```
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pre
d, best_t)))
```

```
Test confusion matrix
[[ 3473  1986]
 [ 9515 21078]]
```

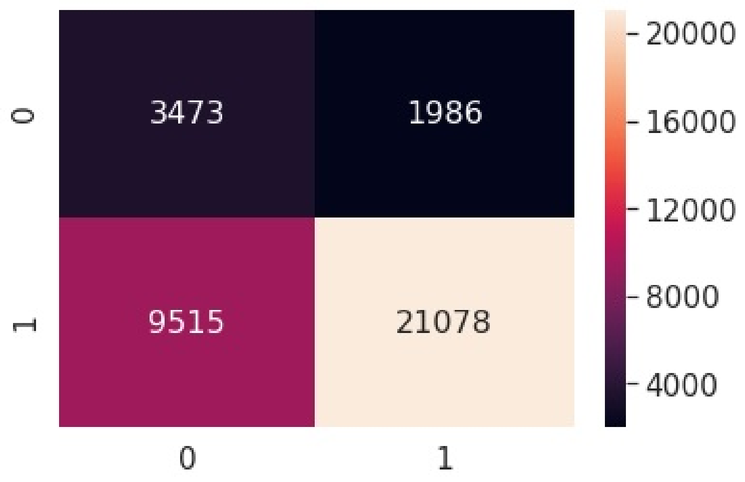
In [139]:

```
print("Test data confusion matrix")

confusion_matrix_df_test = pd.DataFrame(confusion_matrix(y_te
st, predict_with_best_t(y_test_pred, best_t)), range(2),range
(2))
sns.set(font_scale=1.4)#for label size
sns.heatmap(confusion_matrix_df_test, annot=True,annot_kws={"
size": 16}, fmt='g')
```

```
Test data confusion matrix
```

Out[139]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f
2a2f68d198>
```


3473
1986
9515
21078
0
1
0
1
20000
16000
12000
8000
4000

Conclusion

In [143]:

```
# http://zetcode.com/python/prettytable/

from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable
 using: pip3 install prettytable

x = PrettyTable()
x.field_names = ["Vectorizer", "Model", " Best Penalty"," Bes
t Alpha:Hyper Parameter", "Test-AUC"]

x.add_row(["BOW", "Linear SVM","L2 ", 0.01, 0.7])
x.add_row(["TFIDF", "Linear SVM","L1 ", 0.0001, 0.709])
x.add_row(["AVG W2V", "Linear SVM", "L2 ", 0.0001, 0.697 ])
x.add_row(["TFIDF W2V", "Linear SVM", "L2", 0.01 , 0.694 ])
x.add_row(["TRUNCATED SVD", "Linear SVM","L2" , 0.0001, 0.713
])



print(x)
```

```
+---------------+------------+---------------+
-----------------------------+----------+
|   Vectorizer  |   Model    |  Best Penalty |
  Best Alpha:Hyper Parameter | Test-AUC |
+---------------+------------+---------------+
-----------------------------+----------+
|      BOW      | Linear SVM |      L2       |
             0.01            |   0.7    |
|     TFIDF     | Linear SVM |      L1       |
             0.0001           |  0.709   |
|    AVG W2V    | Linear SVM |      L2       |
             0.0001           |  0.697   |
|   TFIDF W2V   | Linear SVM |       L2      |
              0.01            |  0.694   |
| TRUNCATED SVD | Linear SVM |       L2      |
             0.0001           |  0.713   |
+---------------+------------+---------------+
-----------------------------+----------+
```