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	De
`project_id`	A unique identifier for the proposed project. **Example:**
`project_title`	Title of the project. **E** • `Art Will Make You` • `First Gr`
`project_grade_category`	Grade level of students for which the project is targeted. One of the enumerate • `Grades` • `Gra` • `Gra` • `Grac`
`project_subject_categories`	One or more (comma-separated) subject categories for the projec following enumerated list • `Applied L` • `Care &` • `Health &` • `History &` • `Literacy & L` • `Math &` • `Music & T` • `Specia` • **Ex** • `Music & T` • `Literacy & Language, Math &`
`school_state`	State where school is located ([Two-letter U.S. po (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Posta](https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Posta) **Exar**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for th **Ex** • `L` • `Literature & Writing, Social S`
`project_resource_summary`	An explanation of the resources needed for the project. **E** • `My students need hands on literacy materials to sensory`
`project_essay_1`	First applicat
`project_essay_2`	Second applicat
`project_essay_3`	Third applicat
`project_essay_4`	Fourth applicat
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `201 12:43`
`teacher_id`	A unique identifier for the teacher of the proposed project. `bdf8baa8fedef6bfeec7ae4ff`

Feature	De
teacher_prefix	Teacher's title. One of the following enumerate • • • • • • T
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the sam **Exa**

[*] 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

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 [39]:

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

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [40]:

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

In [41]:

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

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

In [42]:

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

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

Out[42]:

	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 [43]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [44]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47
301924/4084039

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

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

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

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

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

1.3 Text preprocessing

In [45]:

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

In [46]:

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

Out[46]:

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

In [47]:

```
#### 1.4.2.3 Using Pretrained Models: TFIDF weighted W2V
```

In [48]:

```
#to drop a row having nan https://stackoverflow.com/questions/13413590
project_data=project_data.dropna(subset=['teacher_prefix'])
```

In [49]:

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

```
My students are English learners that are working on English as their seco
nd or third languages. We are a melting pot of refugees, immigrants, and n
ative-born Americans bringing the gift of language to our school. \r\n\r\n
We have over 24 languages represented in our English Learner program with 
students at every level of mastery.  We also have over 40 countries repres
ented with the families within our school.  Each student brings a wealth o
f knowledge and experiences to us that open our eyes to new cultures, beli
efs, and respect.\"The limits of your language are the limits of your worl
d.\"-Ludwig Wittgenstein  Our English learner's have a strong support syst
em at home that begs for more resources.  Many times our parents are learn
ing to read and speak English along side of their children.  Sometimes thi
s creates barriers for parents to be able to help their child learn phonet
ics, letter recognition, and other reading skills.\r\n\r\nBy providing the
se dvd's and players, students are able to continue their mastery of the E
nglish language even if no one at home is able to assist.  All families wi
th students within the Level 1 proficiency status, will be a offered to be
a part of this program.  These educational videos will be specially chosen
by the English Learner Teacher and will be sent home regularly to watch.  
The videos are to help the child develop early reading skills.\r\n\r\nPare
nts that do not have access to a dvd player will have the opportunity to c
heck out a dvd player to use for the year.  The plan is to use these video
s and educational dvd's for the years to come for other EL students.\r\nna
nnan
==================================================
The 51 fifth grade students that will cycle through my classroom this year
all love learning, at least most of the time. At our school, 97.3% of the 
students receive free or reduced price lunch. Of the 560 students, 97.3% a
re minority students. \r\nThe school has a vibrant community that loves to
get together and celebrate. Around Halloween there is a whole school parad
e to show off the beautiful costumes that students wear. On Cinco de Mayo 
we put on a big festival with crafts made by the students, dances, and gam
es. At the end of the year the school hosts a carnival to celebrate the ha
rd work put in during the school year, with a dunk tank being the most pop
ular activity.My students will use these five brightly colored Hokki stool
s in place of regular, stationary, 4-legged chairs. As I will only have a 
total of ten in the classroom and not enough for each student to have an i
ndividual one, they will be used in a variety of ways. During independent 
reading time they will be used as special chairs students will each use on
occasion. I will utilize them in place of chairs at my small group tables 
during math and reading times. The rest of the day they will be used by th
e students who need the highest amount of movement in their life in order 
to stay focused on school.\r\n\r\nWhenever asked what the classroom is mis
sing, my students always say more Hokki Stools. They can't get their fill 
of the 5 stools we already have. When the students are sitting in group wi
th me on the Hokki Stools, they are always moving, but at the same time do
ing their work. Anytime the students get to pick where they can sit, the H
okki Stools are the first to be taken. There are always students who head 
over to the kidney table to get one of the stools who are disappointed as 
there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 
7 hours a day. The Hokki stools will be a compromise that allow my student
s to do desk work and move at the same time. These stools will help studen
ts to meet their 60 minutes a day of movement by allowing them to activate
their core muscles for balance while they sit. For many of my students, th
ese chairs will take away the barrier that exists in schools for a child w
ho can't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment w
ith plain walls, rows of desks, and a teacher in front of the room? A typi
cal day in our room is nothing like that. I work hard to create a warm inv
iting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed r
```

```
aces in Arkansas.\r\nThey attend a Title I school, which means there is a
high enough percentage of free and reduced-price lunch to qualify. Our sch
ool is an \"open classroom\" concept, which is very unique as there are no
walls separating the classrooms. These 9 and 10 year-old students are very
eager learners; they are like sponges, absorbing all the information and e
xperiences and keep on wanting more.With these resources such as the comfy
red throw pillows and the whimsical nautical hanging decor and the blue fi
sh nets, I will be able to help create the mood in our classroom setting t
o be one of a themed nautical environment. Creating a classroom environmen
t is very important in the success in each and every child's education. Th
e nautical photo props will be used with each child as they step foot into
our classroom for the first time on Meet the Teacher evening. I'll take pi
ctures of each child with them, have them developed, and then hung in our
classroom ready for their first day of 4th grade.  This kind gesture will
set the tone before even the first day of school! The nautical thank you c
ards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help m
e to help make our classroom a fun, inviting, learning environment from da
y one.\r\n\r\nIt costs lost of money out of my own pocket on resources to
get our classroom ready. Please consider helping with this project to make
our new school year a very successful one. Thank you!nannan
==================================================
My wonderful students are 3, 4, and 5 years old.  We are located in a smal
l town outside of Charlotte, NC.  All of my 22 students are children of sc
hool district employees.\r\nMy students are bright, energetic, and they lo
ve to learn!  They love hands-on activities that get them moving.  Like mo
st preschoolers, they enjoy music and creating different things. \r\nAll o
f my students come from wonderful families that are very supportive of our
classroom.  Our parents enjoy watching their children's growth as much as
we do!These materials will help me teach my students all about the life cy
cle of a butterfly.  We will watch as the Painted Lady caterpillars grow b
igger and build their chrysalis.  After a few weeks they will emerge from
the chrysalis as beautiful butterflies!  We already have a net for the chr
ysalises, but we still need the caterpillars and feeding station.\r\nThis
will be an unforgettable experience for my students.  My student absolutel
y love hands-on materials.  They learn so much from getting to touch and m
anipulate different things.  The supporting materials I have selected will
help my students understand the life cycle through exploration.nannan
==================================================
The students in my classroom are learners, readers, writers, explorers, sc
ientists, and mathematicians! The potential in these first graders is endl
ess! Each day they come in grinning from ear-to-ear and ready to learn mor
e. \r\nI choose curriculum that is real and relevant to the students, but
it will also prepare them for their futures. These kids are encouraged to
investigate concepts that are exciting for them and I hope we can keep thi
s momentum going! These kids deserve the best, please help me give that to
them! Thank you! :)These kits include a wide variety of science, technolog
y, engineering, and mechanics for my students to dive into at the beginnin
g of the year. I want them to hit the ground running this upcoming year an
d these kits always encourage high interest.\r\nWho wouldn't want to build
their own roller coaster, design a car, or even think critically to make a
bean bag bounce as far as it can go?? These kits will also shows students
potential careers that they may have never heard of before!\r\nAny donatio
ns would be greatly appreciated and my students will know exactly who to t
hank for them!nannan
==================================================
```

In [50]:

```
# 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 [51]:

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

```
My wonderful students are 3, 4, and 5 years old.  We are located in a smal
l town outside of Charlotte, NC.  All of my 22 students are children of sc
hool district employees.\r\nMy students are bright, energetic, and they lo
ve to learn!  They love hands-on activities that get them moving.  Like mo
st preschoolers, they enjoy music and creating different things. \r\nAll o
f my students come from wonderful families that are very supportive of our
classroom.  Our parents enjoy watching their children is growth as much as
we do!These materials will help me teach my students all about the life cy
cle of a butterfly.  We will watch as the Painted Lady caterpillars grow b
igger and build their chrysalis.  After a few weeks they will emerge from
the chrysalis as beautiful butterflies!  We already have a net for the chr
ysalises, but we still need the caterpillars and feeding station.\r\nThis
will be an unforgettable experience for my students.  My student absolutel
y love hands-on materials.  They learn so much from getting to touch and m
anipulate different things.  The supporting materials I have selected will
help my students understand the life cycle through exploration.nannan
==================================================
```

In [52]:

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

```
My wonderful students are 3, 4, and 5 years old.  We are located in a smal
l town outside of Charlotte, NC.  All of my 22 students are children of sc
hool district employees.  My students are bright, energetic, and they love
to learn!  They love hands-on activities that get them moving.  Like most
preschoolers, they enjoy music and creating different things.   All of my
students come from wonderful families that are very supportive of our clas
sroom.  Our parents enjoy watching their children is growth as much as we
do!These materials will help me teach my students all about the life cycle
of a butterfly.  We will watch as the Painted Lady caterpillars grow bigge
r and build their chrysalis.  After a few weeks they will emerge from the
chrysalis as beautiful butterflies!  We already have a net for the chrysal
ises, but we still need the caterpillars and feeding station.  This will b
e an unforgettable experience for my students.  My student absolutely love
hands-on materials.  They learn so much from getting to touch and manipula
te different things.  The supporting materials I have selected will help m
y students understand the life cycle through exploration.nannan
```

In [53]:

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

```
My wonderful students are 3 4 and 5 years old We are located in a small to
wn outside of Charlotte NC All of my 22 students are children of school di
strict employees My students are bright energetic and they love to learn T
hey love hands on activities that get them moving Like most preschoolers t
hey enjoy music and creating different things All of my students come from
wonderful families that are very supportive of our classroom Our parents e
njoy watching their children is growth as much as we do These materials wi
ll help me teach my students all about the life cycle of a butterfly We wi
ll watch as the Painted Lady caterpillars grow bigger and build their chry
salis After a few weeks they will emerge from the chrysalis as beautiful b
utterflies We already have a net for the chrysalises but we still need the
caterpillars and feeding station This will be an unforgettable experience
for my students My student absolutely love hands on materials They learn s
o much from getting to touch and manipulate different things The supportin
g materials I have selected will help my students understand the life cycl
e through exploration nannan
```

In [54]:

```
# 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'r
e", "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', 't
hey', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "th
at'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
d', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as'
, 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through'
, 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'ov
er', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'an
y', '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", 'no
w', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'migh
tn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'w
asn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [55]:

```
# 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 stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109245/109245 [01:13<00:00, 1485.18it/s]
```

In [56]:

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

Out[56]:

```
'my wonderful students 3 4 5 years old we located small town outside charl
otte nc all 22 students children school district employees my students bri
ght energetic love learn they love hands activities get moving like presch
oolers enjoy music creating different things all students come wonderful f
amilies supportive classroom our parents enjoy watching children growth mu
ch these materials help teach students life cycle butterfly we watch paint
ed lady caterpillars grow bigger build chrysalis after weeks emerge chrysa
lis beautiful butterflies we already net chrysalises still need caterpilla
rs feeding station this unforgettable experience students my student absol
utely love hands materials they learn much getting touch manipulate differ
ent things the supporting materials i selected help students understand li
fe cycle exploration nannan'
```

1.4 Preprocessing of `project_title`

In [57]:

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

```
100%|██████████| 109245/109245 [00:03<00:00, 32797.97it/s]
```

1.5 Preparing data for models

In [58]:

```
project_data.columns
```

Out[58]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_ti
tle',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

```
       - school_state : categorical data
       - clean_categories : categorical data
       - clean_subcategories : categorical data
       - project_grade_category : categorical data
       - teacher_prefix : categorical data

       - project_title : text data
       - text : text data
       - project_resource_summary: text data (optinal)

       - quantity : numerical (optinal)
       - teacher_number_of_previously_posted_projects : numerical
       - price : numerical
```

1.5.1 Vectorizing Categorical data

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

In [59]:

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

```
['AppliedLearning', 'Literacy_Language', 'Health_Sports', 'History_Civic
s', 'Music_Arts', 'SpecialNeeds', 'Warmth', 'Math_Science', 'Care_Hunger']
Shape of matrix after one hot encodig  (109245, 9)
```

In [60]:

```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Extracurricular', 'History_Geography', 'FinancialLiteracy', 'NutritionEd
ucation', 'TeamSports', 'SocialSciences', 'ForeignLanguages', 'Civics_Gove
rnment', 'VisualArts', 'PerformingArts', 'EnvironmentalScience', 'AppliedS
ciences', 'ParentInvolvement', 'CharacterEducation', 'College_CareerPrep',
'CommunityService', 'Health_Wellness', 'Gym_Fitness', 'Other', 'Mathematic
s', 'Health_LifeScience', 'ESL', 'Music', 'EarlyDevelopment', 'Literacy',
'Literature_Writing', 'Warmth', 'SpecialNeeds', 'Economics', 'Care_Hunge
r']
Shape of matrix after one hot encodig  (109245, 30)
```

In [61]:

```
vectorizer = CountVectorizer()
vectorizer.fit(project_data['school_state'].values)

# we use the fitted CountVectorizer to convert the text to vector
state_ohe = vectorizer.transform(project_data['school_state'].values)

print("After vectorizations")
print(state_ohe.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(109245, 51)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi',
'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'm
o', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'o
k', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'w
i', 'wv', 'wy']
==========================================================================
==========================
```

In [62]:

```
vectorizer.fit(project_data['teacher_prefix'].values)

# we use the fitted CountVectorizer to convert the text to vector
teacher_ohe = vectorizer.transform(project_data['teacher_prefix'].values)

print("After vectorizations")
print(teacher_ohe.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(109245, 5)
['dr', 'mr', 'mrs', 'ms', 'teacher']
==========================================================================
==========================
```

In [63]:

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


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

#https://thispointer.com/different-ways-to-remove-a-key-from-dictionary-in-python/
if "Grades" in sorted_grade_dict:
    del sorted_grade_dict["Grades"]


#Vectorizing Categorical data:project_grade_category

# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(vocabulary=list(sorted_grade_dict.keys()), lowercase=False
, binary=True)
vectorizer.fit(project_data['project_grade_category'].values)
print(vectorizer.get_feature_names())


grade_ohe = vectorizer.transform(project_data['project_grade_category'].values)
print(grade_ohe.shape)
```

```
['6-8', 'PreK-2', '3-5', '9-12']
(109245, 4)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [64]:

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

```
Shape of matrix after one hot encodig  (109245, 16623)
```

1.5.2.2 TFIDF vectorizer

In [65]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109245, 16623)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [66]:

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

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

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

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

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

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

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

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


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

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


'''
```

Out[66]:

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/3823034
9/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Mode
l")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    f
or line in tqdm(f):\n        splitLine = line.split()\n        word = spli
tLine[0]\n        embedding = np.array([float(val) for val in splitLine
[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," w
ords loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.
txt\')\n\n# ============================\nOutput:\n    \nLoading Glove Mod
el\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ====
========================\n\nwords = []\nfor i in preproced_texts:\n    wor
ds.extend(i.split(\' \'))\n\nfor i in preproced_titles:\n    words.extend
(i.split(\' \'))\nprint("all the words in the coupus", len(words))\nwords
= set(words)\nprint("the unique words in the coupus", len(words))\n\ninter
_words = set(model.keys()).intersection(words)\nprint("The number of words
that are present in both glove vectors and our coupus",       len(inter_wo
rds),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpu
s = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in word
s_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length",
len(words_courpus))\n\n\n# stronging variables into pickle files python: h
ttp://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-
python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n
pickle.dump(words_courpus, f)\n\n\n'
```

In [67]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-p
ickle-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 [68]:

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

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

```
100%|██████████| 109245/109245 [00:45<00:00, 2407.42it/s]

109245
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [69]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [70]:

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

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

```
100%|██████████| 109245/109245 [04:55<00:00, 369.29it/s]

109245
300
```

1.5.3 Vectorizing Numerical features

In [71]:

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

In [72]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.pr
eprocessing.StandardScaler.html
from sklearn.preprocessing import StandardScaler

# price_standardized = standardScalar.fit(project_data['price'].values)
# this will rise the error
# ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ...
 399.    287.73   5.5 ].
# Reshape your data either using array.reshape(-1, 1)

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

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

```
Mean : 298.1152448166964 ,Standard deviation : 367.49642545627506
```

In [73]:

```
price_standardized
```

Out[73]:

```
array([[-0.39052147],
       [ 0.00240752],
       [ 0.5952024 ],
       ...,
       [-0.1582471 ],
       [-0.61242839],
       [-0.51215531]])
```

1.5.4 Merging all the above features

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

In [74]:

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(109245, 9)
(109245, 30)
(109245, 16623)
(109245, 1)
```

In [75]:

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

Out[75]:

```
(109245, 16663)
```

Computing Sentiment Scores

In [76]:

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

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

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest
students with the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses and multi
ple intelligences i use a wide range\
of techniques to help all my students succeed students in my class come from a variety
 of different backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans our school
is a caring community of successful \
learners which can be seen through collaborative student project based learning in and
 out of the classroom kindergarteners \
in my class love to work with hands on materials and have many different opportunities
 to practice a skill before it is\
mastered having the social skills to work cooperatively with friends is a crucial aspec
t of the kindergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition 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 important math and writing concep
ts while cooking delicious healthy \
food for snack time my students will have a grounded appreciation for the work that wen
t into making the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodies this p
roject would expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to make hom
emade applesauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also create o
ur own cookbooks to be printed and \
shared with families students will gain math and literature skills as well as a life lo
ng enjoyment for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

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

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975
```

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

In [77]:

```
#computing sentiment scores
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

neg1 = []
neu1 = []
pos1 = []
compound1 = []

sid = SentimentIntensityAnalyzer()

for index, row in project_data.iterrows():
    for_sentiment = row['essay']
    ss = sid.polarity_scores(for_sentiment)
    for k in ss:
        if('neg'==k):
            neg1.append(ss[k])
        if(k=='neu'):
            neu1.append(ss[k])
        if(k=='pos'):
            pos1.append(ss[k])
        if(k=='compound'):
            compound1.append(ss[k])
```

In [78]:

```
project_data['neg'] = neg1
project_data['neu'] = neu1
project_data['pos'] = pos1
project_data['compound'] = compound1
```

In [79]:

```
# counting number of words in essay text
no_of_words = []
for index, row in project_data.iterrows():
    words = row['essay']
    res = len(words.split())
    no_of_words.append(res)

project_data['no_of_words_in_essay'] = no_of_words
```

In [80]:

```
# counting number of words in title text
no_of_words = []
for index, row in project_data.iterrows():
    words = row['project_title']
    res = len(words.split())
    no_of_words.append(res)

project_data['no_of_words_in_title'] = no_of_words
```

Normalizing the numerical features: Price

In [81]:

```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(project_data['price'].values.reshape(-1,1))

price_norm = normalizer.transform(project_data['price'].values.reshape(-1,1))

print("After vectorizations")
print(price_norm.shape)
```

```
After vectorizations
(109245, 1)
```

Normalizing the numerical features: teacher_number_of_previously_posted_projects

In [82]:

```
normalizer.fit(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

tnppp_norm = normalizer.transform(project_data['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

print("After vectorizations")
print(tnppp_norm.shape)
```

```
After vectorizations
(109245, 1)
```

Normalizing the numerical features: quantity

In [83]:

```
normalizer.fit(project_data['quantity'].values.reshape(-1,1))

quantity_norm = normalizer.transform(project_data['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(quantity_norm.shape)
```

```
After vectorizations
(109245, 1)
```

Normalizing the numerical features: sentiment score's

In [84]:

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

neg_norm = normalizer.transform(project_data['neg'].values.reshape(-1,1))

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

neu_norm = normalizer.transform(project_data['neu'].values.reshape(-1,1))

normalizer.fit(project_data['pos'].values.reshape(-1,1))

pos_norm = normalizer.transform(project_data['pos'].values.reshape(-1,1))

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

compound_norm = normalizer.transform(project_data['compound'].values.reshape(-1,1))

print("After vectorizations")
print(neg_norm.shape)
print(neu_norm.shape)
print(pos_norm.shape)
print(compound_norm.shape)
```

```
After vectorizations
(109245, 1)
(109245, 1)
(109245, 1)
(109245, 1)
```

Normalizing the numerical features: no_of_words_in_title

In [85]:

```
normalizer.fit(project_data['no_of_words_in_title'].values.reshape(-1,1))

no_of_words_in_title_norm = normalizer.transform(project_data['no_of_words_in_title'].values.reshape(-1,1))
print("After vectorizations")
print(no_of_words_in_title_norm.shape)
```

```
After vectorizations
(109245, 1)
```

Normalizing the numerical features: no_of_words_in_essay

In [86]:

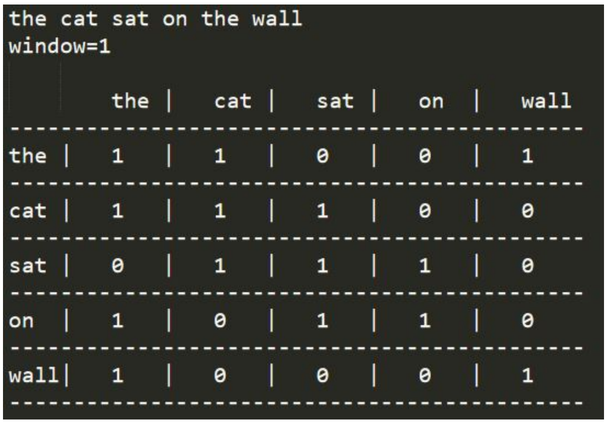
```
normalizer.fit(project_data['no_of_words_in_essay'].values.reshape(-1,1))

no_of_words_in_essay_norm = normalizer.transform(project_data['no_of_words_in_essay'].values.reshape(-1,1))
print("After vectorizations")
print(no_of_words_in_essay_norm.shape)
```

```
After vectorizations
(109245, 1)
```

Assignment 11: TruncatedSVD

- step 1 Select the top 2k words from essay text and project_title (concatinate essay text with project title and then find the top 2k words) based on their `idf_` (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) values
- step 2 Compute the co-occurance matrix with these 2k words, with window size=5 (ref (https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/))



```
the cat sat on the wall
window=1

           the |   cat |   sat |   on  |   wall
----------------------------------------------
the |   1   |   1   |   0   |   0   |   1
----------------------------------------------
cat |   1   |   1   |   1   |   0   |   0
----------------------------------------------
sat |   0   |   1   |   1   |   1   |   0
----------------------------------------------
on  |   1   |   0   |   1   |   1   |   0
----------------------------------------------
wall|   1   |   0   |   0   |   0   |   1
----------------------------------------------
```

- step 3 Use TruncatedSVD (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html) on calculated co-occurance matrix and reduce its dimensions, choose the number of components (`n_components`) using elbow method (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/pca-code-example-using-non-visualization/)

> - The shape of the matrix after TruncatedSVD will be 2000*n, i.e. each row represents a vector form of the corresponding word.
> - Vectorize the essay text and project titles using these word vectors. (while vectorizing, do ignore all the words which are not in top 2k words)

- step 4 Concatenate these truncatedSVD matrix, with the matrix with features
    - **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
    - **word vectors calculated in step 3** : numerical data
- step 5: Apply GBDT on matrix that was formed in step 4 of this assignment, **DO REFER THIS BLOG: XGBOOST DMATRIX (https://www.kdnuggets.com/2017/03/simple-xgboost-tutorial-iris-dataset.html)**
- **step 6:Hyper parameter tuning (Consider any two hyper parameters)**
    - **Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value**
    - **Find the best hyper paramter using k-fold cross validation or simple cross validation data**
    - **Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning**

In [119]:

```
import sys
import math

import numpy as np
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_auc_score

# you might need to install this one
import xgboost as xgb

class XGBoostClassifier():
    def __init__(self, num_boost_round=10, **params):
        self.clf = None
        self.num_boost_round = num_boost_round
        self.params = params
        self.params.update({'objective': 'multi:softprob'})

    def fit(self, X, y, num_boost_round=None):
        num_boost_round = num_boost_round or self.num_boost_round
        self.label2num = {label: i for i, label in enumerate(sorted(set(y)))}
        dtrain = xgb.DMatrix(X, label=[self.label2num[label] for label in y])
        self.clf = xgb.train(params=self.params, dtrain=dtrain, num_boost_round=num_boo
st_round, verbose_eval=1)

    def predict(self, X):
        num2label = {i: label for label, i in self.label2num.items()}
        Y = self.predict_proba(X)
        y = np.argmax(Y, axis=1)
        return np.array([num2label[i] for i in y])

    def predict_proba(self, X):
        dtest = xgb.DMatrix(X)
        return self.clf.predict(dtest)

    def score(self, X, y):
        Y = self.predict_proba(X)[:,1]
        return roc_auc_score(y, Y)

    def get_params(self, deep=True):
        return self.params

    def set_params(self, **params):
        if 'num_boost_round' in params:
            self.num_boost_round = params.pop('num_boost_round')
        if 'objective' in params:
            del params['objective']
        self.params.update(params)
        return self


clf = XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4,)
###################################################################
#                Change from here                                 #
###################################################################
parameters = {
    'num_boost_round': [100, 250, 500],
    'eta': [0.05, 0.1, 0.3],
    'max_depth': [6, 9, 12],
    'subsample': [0.9, 1.0],
    'colsample_bytree': [0.9, 1.0],
}

clf = GridSearchCV(clf, parameters)
X = np.array([[1,2], [3,4], [2,1], [4,3], [1,0], [4,5]])
Y = np.array([0, 1, 0, 1, 0, 1])
clf.fit(X, Y)

# print(clf.grid_scores_)
best_parameters, score, _ = max(clf.grid_scores_, key=lambda x: x[1])
print('score:', score)
for param_name in sorted(best_parameters.keys()):
    print("%s: %r" % (param_name, best_parameters[param_name]))
```

2. TruncatedSVD

2.1 Selecting top 2000 words from `essay` and `project_title`

In [88]:

```
# concatnating essay with text
concat_text = preprocessed_essays + preprocessed_titles
len(concat_text)
```

Out[88]:

```
218490
```

In [89]:

```
#to remove numerical digit in strings and text
import re

def remove(list1):
    list1 = [re.sub(r'[^a-zA-Z ]', '', i) for i in list1]
    return list1

new_concat_text = remove(concat_text)
```

In [90]:

```
#applying tfidf on concatnated text to get idf_score for each unique words in data corp
us
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()
text_tfidf = vectorizer.fit_transform(new_concat_text)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (218490, 57105)
```

In [91]:

```
#objective:to get top important 2k words using idf_score.Since low idf_ means words is important, because this words are frequent in data corpus
idf_score = vectorizer.idf_
#creating dictionary,feature_name as 'key' and idf_score as 'value'
mydict = dict(zip(vectorizer.get_feature_names(), idf_score))
#sorting the dictionary by their idf_score
sorted_x = sorted(mydict.items(), key=lambda kv: kv[1])
#creating pandas dataframe using above sorted dictionary
df = pd.DataFrame(sorted_x, columns=['unique_words','idf_score'])
#creating new dataframe with indicex of 2k
new_df = df.head(2000)
#selecting all values of column:unique_words
top_2k_words = new_df['unique_words'].values
```

In [92]:

```
top_2k_words
```

Out[92]:

```
array(['students', 'nannan', 'school', ..., 'amazed', 'erasers', 'tried'],
      dtype=object)
```

In [93]:

```
#making dictionary of top_2k_words
k=0;
dict_2k_words = {}
for word in top_2k_words:
    dict_2k_words[word] = k
    k+=1
```

2.2 Computing Co-occurance matrix

In [94]:

```
co_occ = np.zeros([2000,2000])
```

In [95]:

```
#calculating co-occurence matrix where new_concat_text is list of whole text corpus can
cating eassay with title.
def cal_occ(sentence):
    total_words = len(sentence.split())
    all_wrds = sentence.split()
    for i,word in enumerate(sentence.split()):
        row_no = dict_2k_words.get(word,-1)
        if(row_no>=0):
            for j in range(max(i-5,0),min(i+6,total_words)):
                col_no = dict_2k_words.get(all_wrds[j],-1)
                if(col_no>=0 and col_no!= row_no):
                    co_occ[row_no][col_no] += 1

from tqdm import tqdm
for sentence in tqdm(new_concat_text):
    cal_occ(sentence)
```

```
100%|██████████| 218490/218490 [02:48<00:00, 1295.80it/s]
```

In [96]:

```
#this is my co-occurence matrix.shape is (2000,2000)
co_occ
```

Out[96]:

```
array([[0.00000e+00, 1.76260e+04, 1.30091e+05, ..., 4.11000e+02,
        3.28000e+02, 2.65000e+02],
       [1.76260e+04, 0.00000e+00, 4.22700e+03, ..., 5.00000e+00,
        2.60000e+01, 7.00000e+00],
       [1.30091e+05, 4.22700e+03, 0.00000e+00, ..., 8.70000e+01,
        5.80000e+01, 8.50000e+01],
       ...,
       [4.11000e+02, 5.00000e+00, 8.70000e+01, ..., 0.00000e+00,
        1.00000e+00, 1.00000e+00],
       [3.28000e+02, 2.60000e+01, 5.80000e+01, ..., 1.00000e+00,
        0.00000e+00, 0.00000e+00],
       [2.65000e+02, 7.00000e+00, 8.50000e+01, ..., 1.00000e+00,
        0.00000e+00, 0.00000e+00]])
```

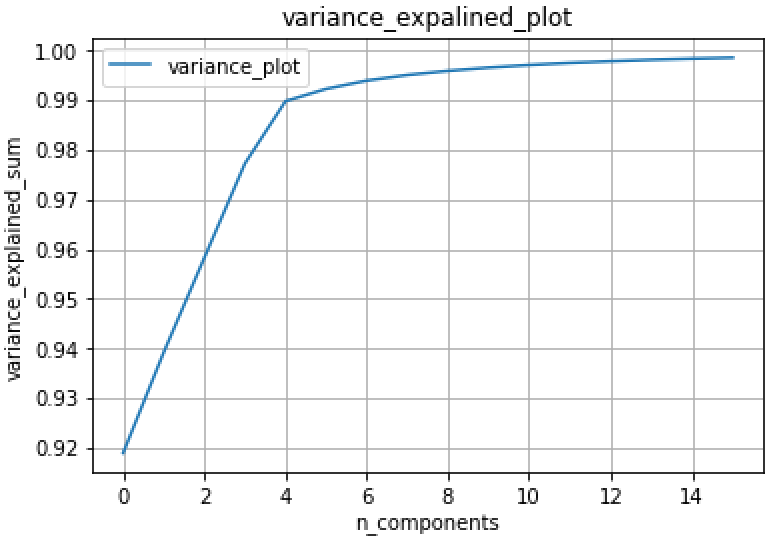
2.3 Applying TruncatedSVD and Calculating Vectors for `essay` and `project_title`

In [97]:

```
from sklearn.decomposition import TruncatedSVD
variance_explained = []
n_components = [5,10,20,40,80,100,120,140,160,180,200,220,240,260,280,300]
for i in n_components:
    svd = TruncatedSVD(n_components=i)
    svd.fit(co_occ)
    variance_explained.append(svd.explained_variance_ratio_.sum())
```

In [98]:

```
plt.plot(variance_explained, label='variance_plot')
plt.legend()
plt.xlabel("n_components")
plt.ylabel("variance_explained_sum")
plt.title("variance_expalined_plot")
plt.grid()
plt.show()
```



In [99]:

```
svd = TruncatedSVD(n_components=4)
svd.fit(co_occ)
w2v=svd.transform(co_occ)
```

In [100]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(4) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        row_no = dict_2k_words.get(word,-1)
        if(row_no>=0):
            vector += w2v[row_no]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

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

```
100%|██████████| 109245/109245 [00:29<00:00, 3667.65it/s]

109245
4
```

In [101]:

```
avg_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is stored in this l
ist
for sentence in tqdm(preprocessed_titles): # for each review/sentence
    vector = np.zeros(4) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        row_no = dict_2k_words.get(word,-1)
        if(row_no>=0):
            vector += w2v[row_no]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles.append(vector)

print(len(avg_w2v_vectors_titles))
print(len(avg_w2v_vectors_titles[0]))
```

```
100%|██████████| 109245/109245 [00:01<00:00, 82504.24it/s]

109245
4
```

2.4 Merge the features from step 3 and step 4

In [102]:

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

Y = project_data['project_is_approved']
X = hstack((categories_one_hot, sub_categories_one_hot, teacher_ohe, state_ohe, grade_o
he, price_norm, quantity_norm,
                  tnppp_norm, neg_norm, neu_norm, pos_norm, compound_norm, no_of_words_
in_essay_norm,
                  no_of_words_in_title_norm, avg_w2v_vectors, avg_w2v_vectors_titles))

print("Final Data matrix")
print(X.shape, Y.shape)
```

```
Final Data matrix
(109245, 116) (109245,)
```

2.5 Apply XGBoost on the Final Features from the above section

https://xgboost.readthedocs.io/en/latest/python/python_intro.html (https://xgboost.readthedocs.io/en/latest/python/python_intro.html)

In [115]:

```
clf = XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4)

parameters = {'max_depth': [2, 4, 6, 8], 'n_estimators': [5, 10, 50, 100]}

clf = GridSearchCV(clf, parameters, return_train_score=True)
clf.fit(X, Y)

X_auc= clf.cv_results_['mean_train_score']
cv_auc = clf.cv_results_['mean_test_score']

#score = clf.best_score_
#print('score:', score)
#best_parameters = clf.best_params_
#for param_name in sorted(best_parameters.keys()):
#    print("%s: %r" % (param_name, best_parameters[param_name]))
```

In [116]:

```
X_auc
```

Out[116]:

```
array([0.57308514, 0.57308514, 0.57308514, 0.57308514, 0.59809531,
       0.59809531, 0.59809531, 0.59809531, 0.63378605, 0.63378605,
       0.63378605, 0.63378605, 0.69826182, 0.69826182, 0.69826182,
       0.69826182])
```

In [108]:

```
#making dataframe for X_auc values with max_depth & n_estimators:
d = {'n_estimators': [5, 10, 50, 100],
    'max_depth = 2': [0.57308514, 0.57308514, 0.57308514, 0.57308514],
    'max_depth = 4': [0.59809531, 0.59809531, 0.59809531, 0.59809531],
    'max_depth = 6': [0.63378605, 0.63378605, 0.63378605, 0.63378605],
    'max_depth = 8': [0.69826182, 0.69826182, 0.69826182, 0.69826182]}
df = pd.DataFrame(d).set_index('n_estimators')
df
```

Out[108]:

	max_depth = 2	max_depth = 4	max_depth = 6	max_depth = 8
n_estimators				
5	0.573085	0.598095	0.633786	0.698262
10	0.573085	0.598095	0.633786	0.698262
50	0.573085	0.598095	0.633786	0.698262
100	0.573085	0.598095	0.633786	0.698262

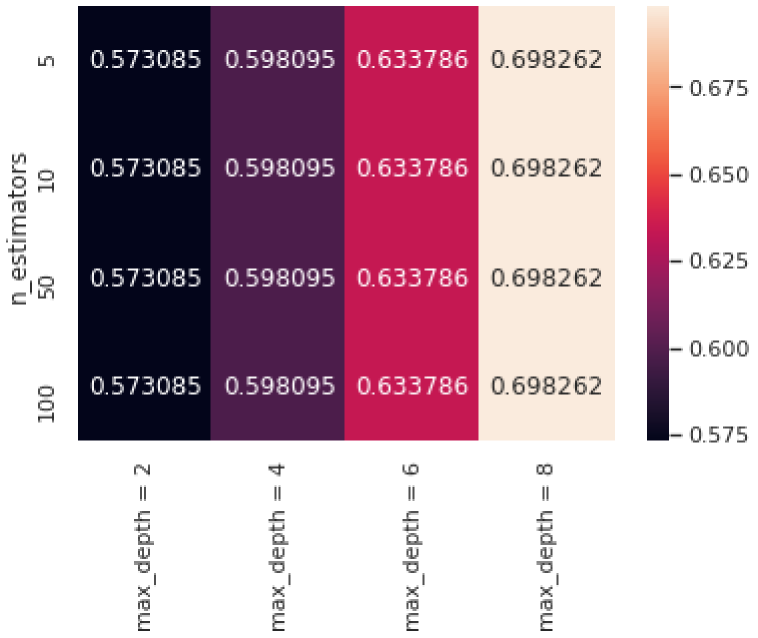
In [109]:

```
#heatmap for X_auc in each cases:
import seaborn as sns; sns.set()

sns.set(font_scale = 1.0)
sns.heatmap(df, annot=True, fmt='g')
```

Out[109]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0955f88630>
```



In [107]:

```
cv_auc
```

Out[107]:

```
array([0.56799726, 0.56799726, 0.56799726, 0.56799726, 0.57560418,
       0.57560418, 0.57560418, 0.57560418, 0.57330059, 0.57330059,
       0.57330059, 0.57330059, 0.5694026 , 0.5694026 , 0.5694026 ,
       0.5694026 ])
```

In [110]:

```
#cv_auc values in each cases:
d = {'n_estimators': [5, 10, 50, 100],
    'max_depth = 2': [0.56799726, 0.56799726, 0.56799726, 0.56799726],
    'max_depth = 4': [0.57560418, 0.57560418, 0.57560418, 0.57560418],
    'max_depth = 6': [0.57330059, 0.57330059, 0.57330059, 0.57330059],
    'max_depth = 8': [0.5694026 , 0.5694026 , 0.5694026, 0.5694026]}
df = pd.DataFrame(d).set_index('n_estimators')
df
```

Out[110]:

	max_depth = 2	max_depth = 4	max_depth = 6	max_depth = 8
n_estimators				
5	0.567997	0.575604	0.573301	0.569403
10	0.567997	0.575604	0.573301	0.569403
50	0.567997	0.575604	0.573301	0.569403
100	0.567997	0.575604	0.573301	0.569403

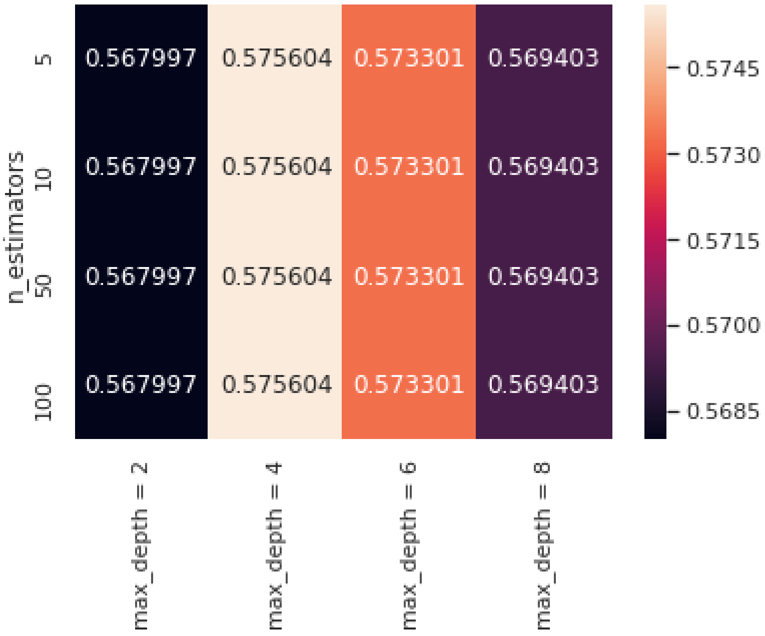
In [111]:

```
#heatmap for cv_auc in each cases:
import seaborn as sns; sns.set()

sns.set(font_scale = 1.0)
sns.heatmap(df, annot=True, fmt='g')
```

Out[111]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0956014160>
```



In [117]:

```
#1.from the heatmap plot we choose max_depth and  n_estimators such that we will have m
aximum AUC on cv data.
#2.Gap between cv_auc and X_auc should be less.

best_max_depth = 4
best_n_estimators = 50
```

In [118]:

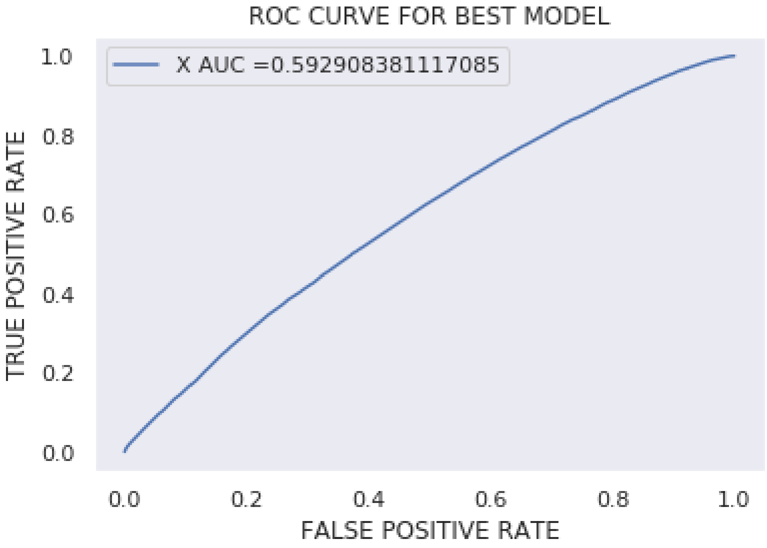
```
# https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#skle
arn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc

clf = XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4, max_depth = 4,
n_estimators = 50)
clf.fit(X, Y)

Y_pred = clf.predict_proba(X)[:,1]

X_fpr, X_tpr, X_thresholds = roc_curve(Y, Y_pred)

plt.plot(X_fpr, X_tpr, label="X AUC ="+str(auc(X_fpr, X_tpr)))
plt.legend()
plt.xlabel("FALSE POSITIVE RATE")
plt.ylabel("TRUE POSITIVE RATE")
plt.title("ROC CURVE FOR BEST MODEL")
plt.grid()
plt.show()
```



3. Conclusion

1.I have learnt how to make word2vector manually.Hard part of this assignment for me is to calculate co-ccurance matrix.But I somehow managed to do it very effectively.

2.Dimensionality of matrix can be reduced with the help of truncated svd.

3.with our own created word2vector matrix,we can vectorize text data.

4.In previous assignment we mainly split our data into train and test/cv.In this assignment we have used xgboost where we don't need to split data in train and test/cv