DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

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

Feature	Descri
project_submitted_datetime	Datetime when project application was submitted. **Example:** 2016-04 12:43:56
teacher_id	A unique identifier for the teacher of the proposed project. **Exa** bdf8baa8fedef6bfeec7ae4ff1c1
teacher_prefix	Teacher's title. One of the following enumerated va • • • • • • Teac
teacher_number_of_previously_posted_projects	Number of project applications previously submitted by the same tea **Exampl**

* 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 [0]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

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

1.1 Reading Data

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

In [0]:
```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

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

In [0]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[0]:

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

1.2 preprocessing of `project_subject_categories`

In [0]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-in-python
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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

In [0]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

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

sub_cat_list = []
for i in sub_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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

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/2289859
5/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

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

1.3 Text preprocessing

In [0]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

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

Out[0]:

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

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

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

My students are English learners that are working on English as their second or third languages. We are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of language to our school. \r\n\r\n We have over 24 languages represented in our English Learner program with students at every level of mastery. We also have over 40 countries represented with the families within our school. Each student brings a wealth of knowledge and ex periences to us that open our eyes to new cultures, beliefs, and respect.\"Th e limits of your language are the limits of your world.\"-Ludwig Wittgenstein Our English learner's have a strong support system at home that begs for more resources. Many times our parents are learning to read and speak English alo ng side of their children. Sometimes this creates barriers for parents to be able to help their child learn phonetics, letter recognition, and other readi ng skills.\r\n\r\nBy providing these dvd's and players, students are able to continue their mastery of the English language even if no one at home is able to assist. All families with students within the Level 1 proficiency status, will be a offered to be a part of this program. These educational videos wil l be specially chosen by the English Learner Teacher and will be sent home re gularly to watch. The videos are to help the child develop early reading ski lls.\r\n\r\nParents that do not have access to a dvd player will have the opp ortunity to check out a dvd player to use for the year. The plan is to use t hese videos and educational dvd's for the years to come for other EL student s.\r\nnannan

==========

The 51 fifth grade students that will cycle through my classroom this year al l love learning, at least most of the time. At our school, 97.3% of the stude nts receive free or reduced price lunch. Of the 560 students, 97.3% are minor ity students. \r\nThe school has a vibrant community that loves to get togeth er and celebrate. Around Halloween there is a whole school parade to show off the beautiful costumes that students wear. On Cinco de Mayo we put on a big f estival with crafts made by the students, dances, and games. At the end of th e year the school hosts a carnival to celebrate the hard work put in during t he school year, with a dunk tank being the most popular activity.My students will use these five brightly colored Hokki stools in place of regular, statio nary, 4-legged chairs. As I will only have a total of ten in the classroom an d not enough for each student to have an individual one, they will be used in a variety of ways. During independent reading time they will be used as speci al 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 th e day they will be used by the students who need the highest amount of moveme nt in their life in order to stay focused on school.\r\n\r\nWhenever asked wh at the classroom is missing, my students always say more Hokki Stools. They c an't get their fill of the 5 stools we already have. When the students are si tting in group with me on the Hokki Stools, they are always moving, but at th e same time doing their work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be taken. There are always students wh o head over to the kidney table to get one of the stools who are disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students t o do desk work and move at the same time. These stools will help students to meet their 60 minutes a day of movement by allowing them to activate their co re muscles for balance while they sit. For many of my students, these chairs will take away the barrier that exists in schools for a child who can't sit s till.nannan

==========

How do you remember your days of school? Was it in a sterile environment with plain walls, rows of desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work hard to create a warm inviting theme

d room for my students look forward to coming to each day.\r\n\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more.With these resources such as the comfy red throw pillows and the whimsical nautical hanging decor and the blue fish nets, I will be able to help create the mood in our classroom setting to be one of a themed nautical environment. Creating a classroom environment is very important in the success in each and every child's education. The nautical photo props will be used with each child as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pictures of each child with them, have them developed, and then hung in our classroom ready for their first day of 4th grade. This kind gesture will set the tone before even the first day of school! The nautical thank you cards will be used throughout the year by the students as they create thank you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of my own pocket on resources to get our classroom ready. Please consider helping with this project to make our new school year a very successful one. Thank you!nannan

==========

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

==========

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

```
is more accessible.nannan
==================================================
```

In [0]:
```
# https://stackoverflow.com/a/47091490/4084039
import re

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

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

In [0]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

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

In [0]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays, cognitive delays, gross/fine motor delays, to autism. They are
eager beavers and always strive to work their hardest working past their limi
tations.     The materials we have are the ones I seek out for my students. I
teach in a Title I school where most of the students receive free or reduced
price lunch.  Despite their disabilities and limitations, my students love co
ming to school and come eager to learn and explore.Have you ever felt like yo
u had ants in your pants and you needed to groove and move as you were in a m
eeting? This is how my kids feel all the time. The want to be able to move as
they learn or so they say.Wobble chairs are the answer and I love then becaus
e they develop their core, which enhances gross motor and in Turn fine motor
skills.   They also want to learn through games, my kids do not want to sit a
nd do worksheets. They want to learn to count by jumping and playing. Physica
l engagement is the key to our success. The number toss and color and shape m
ats can make that happen. My students will forget they are doing work and jus
t have the fun a 6 year old deserves.nannan
```

In [0]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and lan
guage delays cognitive delays gross fine motor delays to autism They are eage
r beavers and always strive to work their hardest working past their limitati
ons The materials we have are the ones I seek out for my students I teach in
a Title I school where most of the students receive free or reduced price lun
ch Despite their disabilities and limitations my students love coming to scho
ol 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 move as they learn or
so they say Wobble chairs are the answer and I love then because they develop
their core which enhances gross motor and in Turn fine motor skills They also
want to learn through games my kids do not want to sit and do worksheets They
want to learn to count by jumping and playing Physical engagement is the key
to our success The number toss and color and shape mats can make that happen
My students will forget they are doing work and just have the fun a 6 year ol
d deserves nannan
```

In [0]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you'
, "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he'
, 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'it
self', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 't
hat', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'becau
se', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'a
ll', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'tha
n', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
d've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
"didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'm
a', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shoul
dn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [0]:
```
# 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%|██████████████████████████████████████████████████████████████████████████████
| 109248/109248 [00:59<00:00, 1828.66it/s]
```

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

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

1.4 Preprocessing of `project_title`

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

1.5 Preparing data for models

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

Out[0]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_titl
e',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

In [0]:
```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

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

In [0]:
```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowe
rcase=False, binary=True)
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcateg
ories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducati
on', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterE
ducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geo
graphy', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'Env
ironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'Spec
ialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (109248, 30)
```

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

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [0]:
```
# We are considering only the words which appeared in at least 10 documents(ro
ws 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  (109248, 16623)
```

In [0]:
```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

1.5.2.2 TFIDF vectorizer

In [0]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.5.2.3 Using Pretrained Models: Avg W2V

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

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

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

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

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

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

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

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


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

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

Out[0]:
```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef loadGloveModel(gloveFile):\n    print ("Loading Glove Model")\n    f = open(gloveFile,\'r\', encoding="utf8")\n    model = {}\n    for line in tqdm(f):\n        splitLine = line.split()\n        word = splitLine[0]\n        embedding = np.array([float(val) for val in splitLine[1:]])\n        model[word] = embedding\n    print ("Done.",len(model)," words loaded!")\n    return model\nmodel = loadGloveModel(\'glove.42B.300d.txt\')\n\n# ==========\nOutput:\n    \nLoading Glove Model\n1917495it [06:32, 4879.69it/s]\nDone. 1917495  words loaded!\n\n# ==========\n\nwords = []\nfor i in preproced_texts:\n    words.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_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")\n\nwords_courpus = {}\nwords_glove = set(model.keys())\nfor i in words:\n    if i in words_glove:\n        words_courpus[i] = model[i]\nprint("word 2 vec length", len(words_courpus))\n\n\n# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/\n\nimport pickle\nwith open(\'glove_vectors\', \'wb\') as f:\n    pickle.dump(words_courpus, f)\n\n\n'
```

In [0]:
```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

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

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

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109248 [00:31<00:00, 3508.17it/s]

109248
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

In [0]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_
)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [0]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in th
is 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/revie
w
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf v
alue((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split
())) # getting the tfidf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)

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

```
100%|██████████████████████████████████████████████████████████████████████
██| 109248/109248 [03:28<00:00, 524.21it/s]

109248
300
```

In [0]:
```
# Similarly you can vectorize for title also
```

1.5.3 Vectorizing Numerical features

In [0]:
```
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'
}).reset_index()
project_data = pd.merge(project_data, price_data, on='id', how='left')
```

In [0]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.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 standard deviation of this data
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")

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

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

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

1.5.4 Merging all the above features

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

In [0]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

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

In [0]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

Out[0]:
```
(109248, 16663)
```

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

Computing Sentiment Scores

In [0]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

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

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the
smallest students with the biggest enthusiasm \
for learning my students learn in many different ways using all of our senses
 and multiple intelligences i use a wide range\
of techniques to help all my students succeed students in my class come from a
variety of different backgrounds which makes\
for wonderful sharing of experiences and cultures including native americans o
ur school is a caring community of successful \
learners which can be seen through collaborative student project based learnin
g in and out of the classroom kindergarteners \
in my class love to work with hands on materials and have many different oppor
tunities to practice a skill before it is\
mastered having the social skills to work cooperatively with friends is a cruc
ial aspect of the kindergarten curriculum\
montana is the perfect place to learn about agriculture and nutrition my stude
nts love to role play in our pretend kitchen\
in the early childhood classroom i have had several kids ask me can we try coo
king with real food i will take their idea \
and create common core cooking lessons where we learn important math and writi
ng concepts while cooking delicious healthy \
food for snack time my students will have a grounded appreciation for the work
that went into making the food and knowledge \
of where the ingredients came from as well as how it is healthy for their bodi
es this project would expand our learning of \
nutrition and agricultural cooking recipes by having us peel our own apples to
make homemade applesauce make our own bread \
and mix up healthy plants from our classroom garden in the spring we will also
create our own cookbooks to be printed and \
shared with families students will gain math and literature skills as well as
 a life long enjoyment for healthy cooking \
nannan'
ss = sid.polarity_scores(for_sentiment)

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

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

```
D:\installed\Anaconda3\lib\site-packages\nltk\twitter\__init__.py:20: UserWar
ning:

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


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

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/))

- 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 [0]:
```
import sys
import math

import numpy as np
from sklearn.grid_search 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_roun
d=num_boost_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]))
```

```
score: 0.8333333333333334
colsample_bytree: 0.9
eta: 0.05
max_depth: 6
num_boost_round: 100
subsample: 0.9
```

2. TruncatedSVD

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

In [ ]:
```
%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.preprocessing import normalize

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

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

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

from tqdm import tqdm_notebook as tqdm1
from tqdm import tqdm
import time
import os

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

from sklearn.model_selection import train_test_split
```

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

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

Text preprocessing(1)

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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-fr
om-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-strin
g-in-python
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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())
```

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

In [ ]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())
my_counter
```

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


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

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

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

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

sub_cat_list = []
for i in sub_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 & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())
```

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

In [ ]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/2289859
5/4084039
from collections import Counter
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())
```

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


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

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

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

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

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

Text preprocessing(2)

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

In [ ]:
```
# Combining all the above statemennts
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)
    sent = re.sub('nannan', '', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

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

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

In [ ]:
```
project_grade_catogories = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflo
w.com/a/47301924/4084039

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

project_grade_cat_list = []
for i in tqdm1(project_grade_catogories):
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Scienc
e", "Warmth", "Care & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on
space "Math & Science"=> "Math","&", "Science"
            j=j.replace('The','') # if we have the words "The" we are going to
replace it with ''(i.e removing 'The')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(emp
ty) ex:"Math & Science"=>"Math&Science"
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tra
iling spaces
        temp = temp.replace('&','_')
    project_grade_cat_list.append(temp.strip())
```

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

In [ ]:
```
project_data.drop(['project_essay_1','project_essay_2','project_essay_3','proj
ect_essay_4'], axis=1, inplace=True)
project_data.head(2)
```

In [ ]:
```
#Replacing Nan's with maximum occured value: https://stackoverflow.com/a/51053
916/8089731
project_data['teacher_prefix'].value_counts().argmax()
project_data.fillna(value=project_data['teacher_prefix'].value_counts().argmax
(),axis=1,inplace=True)
```

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

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

2.2 Make Data Model Ready: encoding numerical, categorical features

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

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

1.4.1 Vectorizing Categorical data

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


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

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


sub_categories_one_hot_train = vectorizer_sub_cat.transform(X_train['clean_sub
categories'].values)
# sub_categories_one_hot_cv = vectorizer_sub_cat.transform(X_cv['clean_subcate
gories'].values)
sub_categories_one_hot_test = vectorizer_sub_cat.transform(X_test['clean_subca
tegories'].values)
print("Shape of matrix after one hot encodig_train ",sub_categories_one_hot_tr
ain.shape)
# print("Shape of matrix after one hot encodig_cv ",sub_categories_one_hot_cv.
shape)
print("Shape of matrix after one hot encodig_test ",sub_categories_one_hot_tes
t.shape)
```

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


school_state_one_hot_train = vectorizer_state.transform(X_train['school_state'
].values)
# school_state_one_hot_cv = vectorizer_state.transform(X_cv['school_state'].va
lues)
school_state_one_hot_test = vectorizer_state.transform(X_test['school_state'].
values)
print("Shape of matrix after one hot encodig_train ",school_state_one_hot_trai
n.shape)
# print("Shape of matrix after one hot encodig_cv ",school_state_one_hot_cv.sh
ape)
print("Shape of matrix after one hot encodig_test ",school_state_one_hot_test.
shape)
```

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

#https://stackoverflow.com/a/39308809/8089731
teacher_prefix_one_hot_train = vectorizer_teacherprefix.transform(X_train['tea
cher_prefix'].values.astype('U'))
# teacher_prefix_one_hot_cv = vectorizer_teacherprefix.transform(X_cv['teacher
_prefix'].values.astype('U'))
teacher_prefix_one_hot_test = vectorizer_teacherprefix.transform(X_test['teach
er_prefix'].values.astype('U'))
print("Shape of matrix after one hot encodig_train ",teacher_prefix_one_hot_tr
ain.shape)
# print("Shape of matrix after one hot encodig_cv ",teacher_prefix_one_hot_cv.
shape)
print("Shape of matrix after one hot encodig_test ",teacher_prefix_one_hot_tes
t[:5,:])
# print(X_train['teacher_prefix'].value_counts())
```

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

#https://stackoverflow.com/a/39308809/8089731
project_grade_category_one_hot_train = vectorizer_projectgrade.transform(X_tra
in['clean_project_grade_category'].values)
# project_grade_category_one_hot_cv = vectorizer_projectgrade.transform(X_cv
['clean_project_grade_category'].values)
project_grade_category_one_hot_test = vectorizer_projectgrade.transform(X_test
['clean_project_grade_category'].values)
print("Shape of matrix after one hot encodig_train ",project_grade_category_on
e_hot_train.shape)
# print("Shape of matrix after one hot encodig_cv ",project_grade_category_one
_hot_cv.shape)
print("Shape of matrix after one hot encodig_test ",project_grade_category_one
_hot_test[:5,:])
```

Vectorizing Numerical features

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

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

# train_text_feature_onehotCoding = normalize(train_text_feature_onehotCoding, axis=0)
# Now standardize the data with above maen and variance.
price_standardized_train = normalize(X_train['price'].values.reshape(-1, 1),axis=0)
# price_standardized_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_standardized_test = normalize(X_test['price'].values.reshape(-1, 1),axis=0)
print(price_standardized_train.shape)
# print(price_standardized_cv.shape)
print(price_standardized_test.shape)
```

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

# 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)

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

# Now standardize the data with above maen and variance.
quantity_standardized_train = normalize(X_train['quantity'].values.reshape(-1, 1),axis=0)
# quantity_standardized_cv = quantity_scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
quantity_standardized_test = normalize(X_test['quantity'].values.reshape(-1, 1),axis=0)
print(quantity_standardized_train.shape)
# print(quantity_standardized_cv.shape)
print(quantity_standardized_test.shape)
```

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

# 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)

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

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized_train = normalize(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1),axis=0)
# teacher_number_of_previously_posted_projects_standardized_cv = teacher_number_of_previously_posted_projects_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
teacher_number_of_previously_posted_projects_standardized_test = normalize(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1),axis=0)
print(teacher_number_of_previously_posted_projects_standardized_train.shape)
# print(teacher_number_of_previously_posted_projects_standardized_cv.shape)
print(teacher_number_of_previously_posted_projects_standardized_test.shape)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [ ]:
```
X_train.head(2)
```

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

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

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

TFIDF Vectorizer on `project_title` (Train,Cv,Test)

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

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

In [1]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

```
C:\Users\LENOVO\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarnin
g: detected Windows; aliasing chunkize to chunkize_serial
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

In [2]:
```
concat_essays_titles= (list(X_train['preprocessed_essays'])+list(X_test['prepr
ocessed_titles']))
```

In [3]:
```
concat_essays_titles[109220]
```

Out[3]:
```
'chromebook research'
```

In [4]:
```
len(concat_essays_titles)
```

Out[4]:
```
109248
```

In [5]:
```
tf_idf_vectorizer = TfidfVectorizer()
tf_idf_vectorizer.fit_transform(concat_essays_titles)
```

Out[5]:
```
<109248x48868 sparse matrix of type '<class 'numpy.float64'>'
        with 7993090 stored elements in Compressed Sparse Row format>
```

In [6]:
```
idf_score = tf_idf_vectorizer.idf_
feature_names  = tf_idf_vectorizer.get_feature_names()
```

In [7]:
```
idf_score_features=[]
for i in range(len(idf_score)):
    idf_score_features.append([idf_score[i],feature_names[i]])
```

In [8]:
```
idf_score_features.sort(reverse=True)
idf_score_features=idf_score_features[:2000]
```

In [9]:
```
# please write all the code with proper documentation, and proper titles for e
ach subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debug
ging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the rea
der
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

2.2 Computing Co-occurance matrix

In [10]:
```
coo_matrix=np.zeros((2000,2000))
window=5
```

In [11]:
```
final_2000_features=[]
for i in range(2000):
    final_2000_features.append(idf_score_features[i][1])
```

In [12]:
```
for sentance in concat_essays_titles:
    word_sen=sentance.split()
    for idss,word in enumerate(word_sen):
        if word in final_2000_features:
            for i in range(max(0,idss-window),min(idss+window,len(word_sen))):
                if word_sen[i] in final_2000_features:
                    coo_matrix[final_2000_features.index(word_sen[i]),final_20
00_features.index(word)]+=1
```

In [ ]:
```
type(coo_matrix)
```

In [15]:
```
# with open('coo_matrix.pkl','wb') as f:
#     pickle.dump(coo_matrix, f)
```

In [13]:
```
with open('coo_matrix.pkl','rb') as f:
    coo_matri = pickle.load(f)
    print(type(coo_matri))
```

```
<class 'numpy.ndarray'>
```

In [14]:
```
coo_matrix = np.array(coo_matri)
```

In [15]:
```
(coo_matrix)
```

Out[15]:
```
array([[1., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 1., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 1., 0., 0.],
       [0., 0., 0., ..., 0., 1., 0.],
       [0., 0., 0., ..., 0., 0., 2.]])
```

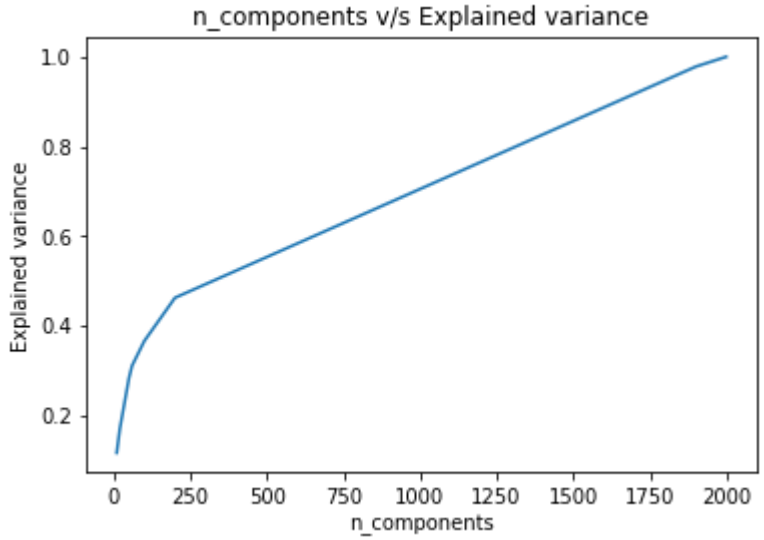
In [ ]:
```

```

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

In [16]:
```
# finding optimal value of n_componenets(n) using truncated svd
from sklearn.decomposition import TruncatedSVD
n_components=[10,20,50,60,100,200,300,400,500,1000,1200,1500,1600,1700,1800,19
00,1999]
explained_variance=[]
for n in n_components:
    svd=TruncatedSVD(n_components=n,random_state=42)
    svd.fit(coo_matrix)
    exvar=svd.explained_variance_ratio_.sum()
    explained_variance.append(exvar)

    print('n_components=',n,'variance=',exvar)
```

```
n_components= 10 variance= 0.1159590813580738
n_components= 20 variance= 0.16923911371858813
n_components= 50 variance= 0.28315252103920313
n_components= 60 variance= 0.31047651235692136
n_components= 100 variance= 0.3656284135288057
n_components= 200 variance= 0.46206687050301337
n_components= 300 variance= 0.49242876027040444
n_components= 400 variance= 0.5227869453600746
n_components= 500 variance= 0.5531439823006257
n_components= 1000 variance= 0.7049379035686646
n_components= 1200 variance= 0.765651384750285
n_components= 1500 variance= 0.8567388230867455
n_components= 1600 variance= 0.8870934317555152
n_components= 1700 variance= 0.9174524165591598
n_components= 1800 variance= 0.9478132776464756
n_components= 1900 variance= 0.9781688846378389
n_components= 1999 variance= 0.9999999999999809
```

In [17]:
```
#plotting curve between n_components and explained variance
plt.plot(n_components, explained_variance)
plt.xlabel('n_components')
plt.ylabel("Explained variance")
plt.title("n_components v/s Explained variance")
plt.show()
```



In [18]:
```
from sklearn.decomposition import TruncatedSVD

tsvd=TruncatedSVD(n_components=1800,random_state=42)
final_coo_matrix=tsvd.fit_transform(coo_matrix)
```

In [19]:
```
final_coo_matrix.shape
```

Out[19]:
```
(2000, 1800)
```

In [20]:
```
final_coo_matrix
```

Out[20]:
```
array([[ 1.32840022e-12,  2.68385593e-12,  3.24824011e-12, ...,
        -1.39555451e-02,  4.56617844e-02,  3.25500740e-02],
       [ 1.34762544e-12, -7.07120759e-12, -9.48902320e-12, ...,
         1.45315793e-02, -8.68778904e-03,  3.49422371e-03],
       [ 8.32975281e-13,  2.55641967e-12, -1.38707945e-11, ...,
         5.35582481e-02, -2.89401347e-02,  2.52105460e-02],
       ...,
       [ 2.99595591e-12,  1.49595751e-12, -1.40947788e-11, ...,
         1.82719563e-02, -2.54953153e-02,  1.33273143e-02],
       [-1.21354773e-12, -1.99786334e-12,  2.71971481e-11, ...,
        -1.58616926e-02, -2.59020772e-02, -2.16188073e-02],
       [-1.47697393e-16,  1.51848033e-16,  8.46502386e-15, ...,
        -2.06532044e-17, -3.66672093e-17,  8.64100661e-18]])
```

In [21]:
```
final_coo_matrix[0]
```

Out[21]:
```
array([ 1.32840022e-12,  2.68385593e-12,  3.24824011e-12, ...,
       -1.39555451e-02,  4.56617844e-02,  3.25500740e-02])
```

In [22]:
```
model = {}
for i in range(len(final_2000_features)):
    model[final_2000_features[i]] = final_coo_matrix[i]
```

In [23]:
```
# model = final_2000_features
glove_words =  set(model.keys())
```

In [24]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_titles_vectors_train = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm1(X_train['preprocessed_titles']): # for each review/sente
nce
    vector = np.zeros(1800) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_titles_vectors_train.append(vector)
```

In [25]:
```
avg_w2v_essays_vectors_train = []; # the avg-w2v for each sentence/review is s
tored in this list
for sentence in tqdm1(X_train['preprocessed_essays']): # for each review/sente
nce
    vector = np.zeros(1800) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_essays_vectors_train.append(vector)
```

In [26]:
```
avg_w2v_titles_vectors_test = []; # the avg-w2v for each sentence/review is st
ored in this list
for sentence in tqdm1(X_test['preprocessed_titles']): # for each review/senten
ce
    vector = np.zeros(1800) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_titles_vectors_test.append(vector)
```

In [27]:
```
avg_w2v_essays_vectors_test = []; # the avg-w2v for each sentence/review is st
ored in this list
for sentence in tqdm1(X_test['preprocessed_essays']): # for each review/senten
ce
    vector = np.zeros(1800) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_essays_vectors_test.append(vector)
```

2.4 Merge the features from step 3 and step 4

Word counts(TITLES)

In [29]:
```
title_wordcount_train = []
title_train = list(X_train['preprocessed_titles'])
for i in tqdm1(title_train):
    b = len(str(i).split())
    title_wordcount_train.append(b)
title_wordcount_train = np.array(title_wordcount_train)

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

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

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

Standardizing Word counts(TITLES)

In [30]:
```
from sklearn.preprocessing import StandardScaler

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

title_wordcount_standardized_train = title_wordcount_scalar.transform(title_wo
rdcount_train.reshape(-1,1))
title_wordcount_standardized_test = title_wordcount_scalar.transform(title_wor
dcount_test.reshape(-1,1))

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

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

Word counts(ESSAYS)

In [31]:
```
essay_wordcount_train = []
essay_train = list(X_train['preprocessed_essays'])
for i in tqdm1(essay_train):
    b = len(str(i).split())
    essay_wordcount_train.append(b)
essay_wordcount_train = np.array(essay_wordcount_train)


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

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

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

Standardizing Word counts(ESSAYS)

In [32]:
```
from sklearn.preprocessing import StandardScaler

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

essay_wordcount_standardized_train = essay_wordcount_scalar.transform(essay_wo
rdcount_train.reshape(-1,1))
essay_wordcount_standardized_test = essay_wordcount_scalar.transform(essay_wor
dcount_test.reshape(-1,1))

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

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

Sentiment scores for each essay

In [ ]:
```
```

In [35]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

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

```
sid = SentimentIntensityAnalyzer()

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

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

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

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


############################################################

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

essay_negscore_test = []
essay_neuscore_test = []
essay_posscore_test = []
essay_compoundscore_test = []
for it in essay_sentscore_test:
    a = it['neg']
    essay_negscore_test.append(a)
    b = it['neu']
    essay_neuscore_test.append(b)
    c = it['pos']
    essay_posscore_test.append(c)
    d = it['compound']
    essay_compoundscore_test.append(d)
```

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

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

```
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /home/dileep_teja3/nltk_data...


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


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

In [36]:
```
final_coo_matrix.shape
```

Out[36]: (2000, 1800)

In [37]:
```
categories_one_hot_train.shape
```

Out[37]: (73196, 9)

In [38]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_tr = hstack((categories_one_hot_train,sub_categories_one_hot_train,school_state_one_hot_train,teacher_prefix_one_hot_train
               ,project_grade_category_one_hot_train,price_standardized_train,quantity_standardized_train
               ,teacher_number_of_previously_posted_projects_standardized_train,essay_negscore_train,essay_neuscore_train,essay_posscore_train
               ,essay_compoundscore_train,title_wordcount_standardized_train
               ,essay_wordcount_standardized_train,avg_w2v_titles_vectors_train,avg_w2v_essays_vectors_train)).tocsr()

X_te = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_state_one_hot_test,teacher_prefix_one_hot_test
               ,project_grade_category_one_hot_test,price_standardized_test,quantity_standardized_test
               ,teacher_number_of_previously_posted_projects_standardized_test,essay_negscore_test,essay_neuscore_test,essay_posscore_test
               ,essay_compoundscore_test,title_wordcount_standardized_test
               ,essay_wordcount_standardized_test,avg_w2v_titles_vectors_test,avg_w2v_essays_vectors_test)).tocsr()

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

```
(73196, 3708) (73196,)
(36052, 3708) (36052,)
====================================================================================================
```

In [ ]:

In [ ]:

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

In [40]:
```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
import time

start_time = time.time()
gbdt = xgb.XGBClassifier(n_jobs=-1,class_weight='balanced')
parameters = {'n_estimators': [10, 100, 500], 'max_depth':[10, 50, 100, 500]}
clf = GridSearchCV(gbdt, parameters, cv= 3, scoring='roc_auc',return_train_sco
re=True)
clf.fit(X_tr, y_train)

train_auc= clf.cv_results_['mean_train_score']
train_auc_std= clf.cv_results_['std_train_score']
cv_auc = clf.cv_results_['mean_test_score']
cv_auc_std= clf.cv_results_['std_test_score']
print("Execution time: " + str((time.time() - start_time)) + ' ms')
```

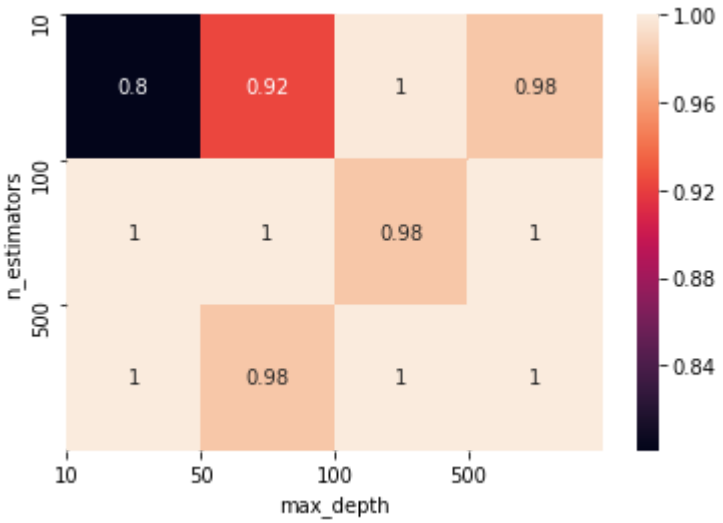
```
Execution time: 1624.1976990699768 ms
```

In [1]:
```
import dill
# dill.dump_session('notebook_env11.db')
dill.load_session('notebook_env11.db')
```

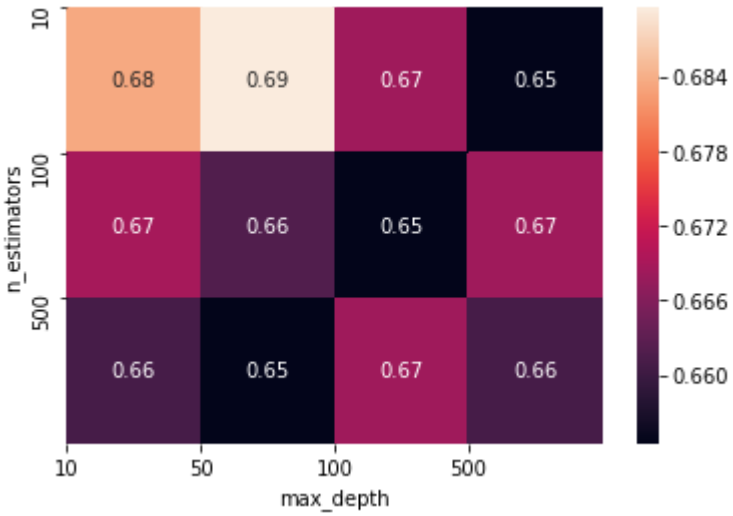
In [2]:
```
train_auc = train_auc.reshape(3,4)
cv_auc = cv_auc.reshape(3,4)
train_auc
cv_auc
```

Out[2]:
```
array([[0.68351461, 0.68952022, 0.66843937, 0.65448579],
       [0.66898794, 0.66197499, 0.65448579, 0.66840205],
       [0.66095329, 0.65448579, 0.66840205, 0.66095329]])
```

In [3]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(train_auc,annot=True)

plt.yticks(np.arange(3), [10, 100, 500])
plt.xticks(np.arange(4), [10, 50, 100, 500])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```

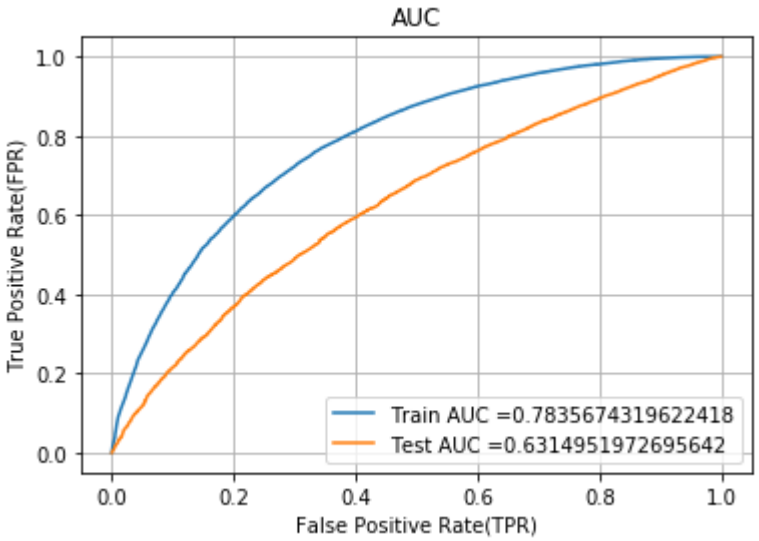


In [4]:
```
import matplotlib.pyplot as plt
# plt.show()

import numpy as np; np.random.seed(0)
import seaborn as sns


sns.heatmap(cv_auc,annot=True)

plt.yticks(np.arange(3), [10, 100, 500])
plt.xticks(np.arange(4), [10, 50, 100, 500])

plt.xlabel('max_depth')
plt.ylabel('n_estimators')


plt.show()
```



In [5]:
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability e
stimates of the positive class
    # not the predicted outputs

    y_data_pred = []
    tr_loop = data.shape[0] - data.shape[0]%1000
    # consider you X_tr shape is 49041, then your cr_loop will be 49041 - 4904
1%1000 = 49000
    # in this for loop we will iterate unti the last 1000 multiplier
    for i in range(0, tr_loop, 1000):
        y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
    # we will be predicting for the last data points
    y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])
    return y_data_pred
```

In [6]:
```
from sklearn.metrics import roc_curve, auc

gbdt = xgb.XGBClassifier(max_depth = 10, n_estimators = 10,n_jobs=-1,class_wei
ght='balanced')
gbdt.fit(X_tr, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estim
ates of the positiveclass
# not the predicted outputs

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

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

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



In [7]:
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

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

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

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

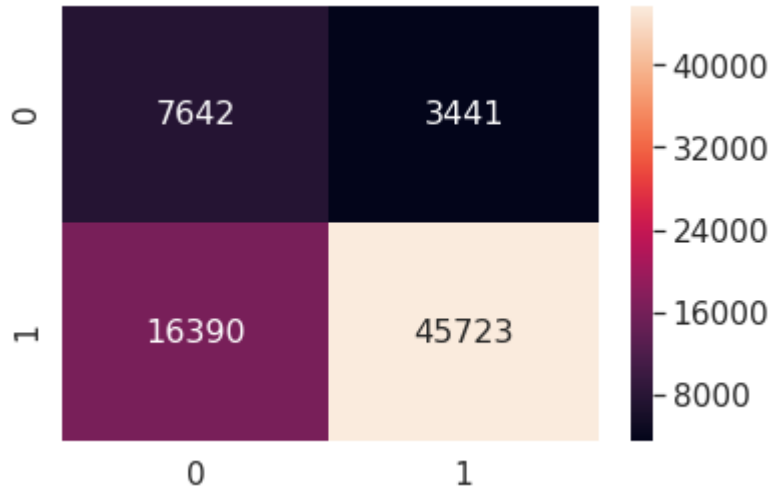
In [8]:
```
print("="*100)
from sklearn.metrics import confusion_matrix
print("Train confusion matrix")
print(confusion_matrix(y_train[:], predict(y_train_pred, tr_thresholds, train_
fpr, train_tpr)))
print("Test confusion matrix")
print(confusion_matrix(y_test[:], predict(y_test_pred, tr_thresholds, test_fpr
, test_tpr)))
```

```
====================================================================================
======================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5075769738266626 for threshold 0.713
[[ 7642  3441]
 [16390 45723]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.35821232052725976 for threshold 0.741
[[ 5316   143]
 [28497  2096]]
```

In [9]:
```
conf_matr_df_train =  pd.DataFrame(confusion_matrix(y_train[:], predict(y_trai
n_pred, tr_thresholds, train_fpr, train_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_train, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.5075769738266626 for threshold 0.713
```

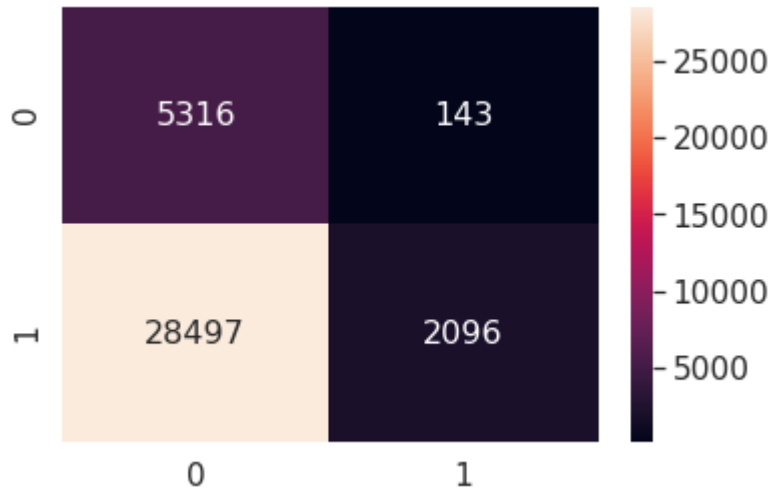
Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48da7fffd0>
```



In [10]:
```
conf_matr_df_test =  pd.DataFrame(confusion_matrix(y_test[:], predict(y_test_p
red, tr_thresholds, test_fpr, test_tpr)))
sns.set(font_scale=1.4)#for label size
sns.heatmap(conf_matr_df_test, annot=True,annot_kws={"size": 16}, fmt='g')
```

```
the maximum value of tpr*(1-fpr) 0.35821232052725976 for threshold 0.741
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f48da769c88>
```



In [ ]:
```

```

3. Conclusion

In [12]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "AUC"]
x.add_row(["AVG W2V", "XGBoost", "Max Depth:10 , n_estimators:10", 0.63])
print(x)
```

```
+------------+---------+--------------------------------+------+
| Vectorizer |  Model  |        Hyper Parameter         | AUC  |
+------------+---------+--------------------------------+------+
|  AVG W2V   | XGBoost | Max Depth:10 , n_estimators:10 | 0.63 |
+------------+---------+--------------------------------+------+
```

In [ ]: