DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	
`project_id`	A unique identifier for the proposed project
`project_title`	Title of t • `Art Wil` •
`project_grade_category`	Grade level of students for which the project is targete • • • •

Feature	
project_subject_categories	One or more (comma-separated) subject categories f following enur • • • • • Li • • • •
school_state	State where school is located (Two- (https://en.wikipedia.org/wiki/List_of_U.S._state_abbrevia
project_subject_subcategories	One or more (comma-separated) subject subcate • • Literature & Writing
project_resource_summary	An explanation of the resources needed for t • My students need hands on literacy mat sen
project_essay_1	F
project_essay_2	Sec
project_essay_3	Tl
project_essay_4	Fou
project_submitted_datetime	Datetime when project application was submitted. **Ex**
teacher_id	A unique identifier for the teacher of the propos bdf8baa8fedef6b
teacher_prefix	Teacher's title. One of the following • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submitted

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?clie
nt_id=947318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.
com&redirect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3
A%2F%2Fwww.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapi
s.com%2Fauth%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.phot
os.readonly%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&
response_type=code (https://accounts.google.com/o/oauth2/auth?client_id=947
318989803-6bn6qk8qdgf4n4g3pfee6491hc0brc4i.apps.googleusercontent.com&redir
ect_uri=urn%3Aietf%3Awg%3Aoauth%3A2.0%3Aoob&scope=email%20https%3A%2F%2Fww
w.googleapis.com%2Fauth%2Fdocs.test%20https%3A%2F%2Fwww.googleapis.com%2Fau
th%2Fdrive%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.photos.readonl
y%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fpeopleapi.readonly&response_t
ype=code)

Enter your authorization code:
··········
Mounted at /content/drive
```

In [2]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

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

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

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

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

from tqdm import tqdm
import os

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

1.1 Reading Data

In [0]:
```
project_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/train_dat
resource_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/resource
```

In [4]:
```
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 (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix'
'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [5]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[5]:

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

1.2 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

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-f
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-stri
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science
        if 'The' in j.split(): # this will split each of the catogory based o
            j=j.replace('The','') # if we have the words "The" we are going t
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(em
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the tr
        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

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science
        if 'The' in j.split(): # this will split each of the catogory based o
            j=j.replace('The','') # if we have the words "The" we are going t
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(em
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the tr
        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/228985
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 [9]:
```
project_data.head(2)
```

Out[9]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	projec
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 [11]:
```
# 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)
```

```
My students are English learners that are working on English as their secon
d or third languages. We are a melting pot of refugees, immigrants, and nat
ive-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 stud
ents at every level of mastery.  We also have over 40 countries represented
with the families within our school.  Each student brings a wealth of knowl
edge and experiences to us that open our eyes to new cultures, beliefs, and
respect.\"The limits of your language are the limits of your world.\"-Ludwi
g Wittgenstein  Our English learner's have a strong support system at home
that begs for more resources.  Many times our parents are learning to read
and speak English along side of their children.  Sometimes this creates bar
riers for parents to be able to help their child learn phonetics, letter re
cognition, and other reading skills.\r\n\r\nBy providing these dvd's and pl
ayers, 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 withi
n the Level 1 proficiency status, will be a offered to be a part of this pr
ogram.  These educational videos will be specially chosen by the English Le
arner Teacher and will be sent home regularly to watch.  The videos are to
help the child develop early reading skills.\r\n\r\nParents that do not hav
e access to a dvd player will have the opportunity to check out a dvd playe
r to use for the year.  The plan is to use these videos and educational dv
d's for the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year
all love learning, at least most of the time. At our school, 97.3% of the s
tudents receive free or reduced price lunch. Of the 560 students, 97.3% are
minority students. \r\nThe school has a vibrant community that loves to get
together and celebrate. Around Halloween there is a whole school parade to
show off the beautiful costumes that students wear. On Cinco de Mayo we put
on a big festival with crafts made by the students, dances, and games. At t
he end of the year the school hosts a carnival to celebrate the hard work p
ut in during the school year, with a dunk tank being the most popular activ
ity.My students will use these five brightly colored Hokki stools in place
of regular, stationary, 4-legged chairs. As I will only have a total of ten
in the classroom and not enough for each student to have an individual one,
they will be used in a variety of ways. During independent reading time the
y will be used as special chairs students will each use on occasion. I will
utilize them in place of chairs at my small group tables during math and re
ading times. The rest of the day they will be used by the students who need
the highest amount of movement in their life in order to stay focused on sc
hool.\r\n\r\nWhenever asked what the classroom is missing, my students alwa
ys say more Hokki Stools. They can't get their fill of the 5 stools we alre
ady have. When the students are sitting in group with me on the Hokki Stool
s, they are always moving, but at the same time doing their work. Anytime t
he students get to pick where they can sit, the Hokki Stools are the first
to be taken. There are always students who head over to the kidney table to
get one of the stools who are disappointed as there are not enough of them.
```

\r\n\r\nWe ask a lot of students to sit for 7 hours a day. The Hokki stools will be a compromise that allow my students to do desk work and move at the same time. These stools will help students to meet their 60 minutes a day o f movement by allowing them to activate their core muscles for balance whil e they sit. For many of my students, these chairs will take away the barrie r that exists in schools for a child who can't sit still.nannan
==========
How do you remember your days of school? Was it in a sterile environment wi th plain walls, rows of desks, and a teacher in front of the room? A typica l day in our room is nothing like that. I work hard to create a warm inviti ng themed room for my students look forward to coming to each day.\r\n\r\nM y class is made up of 28 wonderfully unique boys and girls of mixed races i n Arkansas.\r\nThey attend a Title I school, which means there is a high en ough percentage of free and reduced-price lunch to qualify. Our school is a n \"open classroom\" concept, which is very unique as there are no walls se parating the classrooms. These 9 and 10 year-old students are very eager le arners; they are like sponges, absorbing all the information and experience s 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 imp ortant in the success in each and every child's education. The nautical pho to 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 befo re 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 c lassroom a fun, inviting, learning environment from day one.\r\n\r\nIt cost s lost of money out of my own pocket on resources to get our classroom read y. 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 l anguage delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past thei r 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 fr ee 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 t o 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, m y 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 num ber toss and color and shape mats can make that happen. My students will fo rget they are doing work and just have the fun a 6 year old deserves.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 [13]:
```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and l
anguage delays, cognitive delays, gross/fine motor delays, to autism. They
are eager beavers and always strive to work their hardest working past thei
r 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 fr
ee 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 t
o 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, m
y 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 num
ber toss and color and shape mats can make that happen. My students will fo
rget they are doing work and just have the fun a 6 year old deserves.nannan
==================================================
```

In [14]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and l
anguage delays, cognitive delays, gross/fine motor delays, to autism. They
are eager beavers and always strive to work their hardest working past thei
r limitations.     The materials we have are the ones I seek out for my stu
dents. I teach in a Title I school where most of the students receive free
or reduced price lunch.  Despite their disabilities and limitations, my stu
dents love coming to school and come eager to learn and explore.Have you ev
er felt like you had ants in your pants and you needed to groove and move a
s you were in a meeting? This is how my kids feel all the time. The want to
be able to move as they learn or so they say.Wobble chairs are the answer a
nd I love then because they develop their core, which enhances gross motor
and in Turn fine 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 j
umping and playing. Physical engagement is the key to our success. The numb
er toss and color and shape mats can make that happen. My students will for
get they are doing work and just have the fun a 6 year old deserves.nannan
```

In [15]:
```
#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 l
anguage delays cognitive delays gross fine motor delays to autism They are
eager beavers and always strive to work their hardest working past their li
mitations The materials we have are the ones I seek out for my students I t
each in a Title I school where most of the students receive free or reduced
price lunch Despite their disabilities and limitations my students love com
ing to school and come eager to learn and explore Have you ever felt like y
ou 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 b
ecause 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 s
it and do worksheets They want to learn to count by jumping and playing Phy
sical engagement is the key to our success The number toss and color and sh
ape mats can make that happen My students will forget they are doing work a
nd just have the fun a 6 year old 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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'i
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this',
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'beca
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into',
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'th
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "shoul
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn',
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't",
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shou
            'won', "won't", 'wouldn', "wouldn't"]
```

In [17]:
```
# 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%|██████████| 50000/50000 [00:24<00:00, 2005.87it/s]
```

In [18]:
```
# after preprocesing
preprocessed_essays[20000]
```

Out[18]:
```
'my kindergarten students varied disabilities ranging speech language delay
s cognitive delays gross fine motor delays autism they eager beavers always
strive work hardest working past limitations the materials ones i seek stud
ents i teach title i school students receive free reduced price lunch despi
te disabilities limitations students love coming school come eager learn ex
plore have ever felt like ants pants needed groove move meeting this kids f
eel time the want able move learn say wobble chairs answer i love develop c
ore enhances gross motor turn fine motor skills they also want learn games
kids not want sit worksheets they want learn count jumping playing physical
engagement key success the number toss color shape mats make happen my stud
ents forget work fun 6 year old deserves nannan'
```

In [0]:
```
project_data['essay'] = preprocessed_essays
```

1.4 Preprocessing of `project_title`

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

In [21]:
```
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 50000/50000 [00:01<00:00, 43394.31it/s]
```

In [0]:
```
project_data['project_title'] = preprocessed_titles
```

In [0]:
```
#Preprocessing project_grade_category

#reference link: https://stackoverflow.com/questions/28986489/python-pandas-h

project_data['project_grade_category'] = project_data['project_grade_category
project_data['project_grade_category'] = project_data['project_grade_category
```

1.5 Preparing data for models

In [24]:
```
project_data.columns
```

Out[24]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_tit
le',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay'],
      dtype='object')
```

we are going to consider

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

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

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

1.5.1 Vectorizing Categorical data

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

In [25]:
```
# 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()), lowerca
categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearnin
g', 'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (50000, 9)
```

In [26]:
```
# we use count vectorizer to convert the values into one
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lov
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcate
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', 'NutritionEduca
tion', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Charac
terEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'Histo
ry_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitnes
s', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedScienc
es', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (50000, 30)
```

In [28]:
```
#school state
#Using CountVectorizer to convert values into one hot encoded
vectorizer = CountVectorizer(lowercase=False , binary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())

school_state_one_hot = vectorizer.transform(project_data['school_state'].valu
print('Shape of matrix after one hot encoding', school_state_one_hot.shape)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'I
A', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO',
'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'O
R', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV',
'WY']
Shape of matrix after one hot encoding (50000, 51)
```

In [29]:
```
#teacher_prefix
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'].values.astype('U'))
#While running this i got an error:np.nan is an invalid document, expected by
#I fixed it by using stackoverflow.com
#https://stackoverflow.com/questions/39303912/tfidfvectorizer-in-scikit-learr
print(vectorizer.get_feature_names())


teacher_prefix_one_hot = vectorizer.transform(project_data['teacher_prefix'].
print('Shape of matrix of one hot encoding', teacher_prefix_one_hot.shape)
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher', 'nan']
Shape of matrix of one hot encoding (50000, 6)
```

In [30]:
```
#project_grade_category
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['project_grade_category'].values.astype('U'))
print(vectorizer.get_feature_names())


project_grade_category_one_hot = vectorizer.fit_transform(project_data['proje
print('Shape of matrix of one hot encoding', project_grade_category_one_hot.s
```

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
Shape of matrix of one hot encoding (50000, 4)
```

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

In [31]:
```
# We are considering only the words which appeared in at least 10 documents(r
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  (50000, 12211)
```

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

In [33]:
```
vectorizer = CountVectorizer(min_df=10)
title_bow = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",title_bow.shape)
```

```
Shape of matrix after one hot encodig  (50000, 2039)
```

1.5.2.2 TFIDF vectorizer

In [34]:
```
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  (50000, 12211)
```

In [35]:
```
vectorizer = TfidfVectorizer(min_df=10)
title_tfidf = vectorizer.fit_transform(preprocessed_titles)
print("Shape of matrix after one hot encodig ",title_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (50000, 2039)
```

1.5.2.3 Using Pretrained Models: Avg W2V

In [36]:

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/40846
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 cou
      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/ho

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


'''
```

Out[36]:
```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef (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 (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/ho
# make sure you have the glove_vectors file
with open('/content/drive/My Drive/Colab Notebooks/glove_vectors', 'rb') as f
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [38]:
```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in thi
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%|██████████| 50000/50000 [00:15<00:00, 3268.13it/s]

50000
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 [40]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in t
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/revi
    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
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spli
            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%|██████████| 50000/50000 [01:42<00:00, 489.06it/s]

50000
300
```

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

In [42]:
```
# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors_titles = []; # the avg-w2v for each sentence/review is stor
for sentence in tqdm(preprocessed_titles): # 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/revi
    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
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.spli
            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_titles.append(vector)

print(len(tfidf_w2v_vectors_titles))
print(len(tfidf_w2v_vectors_titles[0]))
```

```
100%|██████████| 50000/50000 [00:01<00:00, 27689.14it/s]

50000
300
```

1.5.3 Vectorizing Numerical features

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

In [44]:
```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/
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 3
# Reshape your data either using array.reshape(-1, 1)

price_scalar = StandardScaler()
price_scalar.fit(project_data['price'].values.reshape(-1,1)) # finding the me
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_s

# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.resh
```

```
Mean : 299.33367619999996, Standard deviation : 378.20927190421384
```

In [45]:
```
price_standardized
```

Out[45]:
```
array([[-0.38268146],
       [-0.00088225],
       [ 0.57512161],
       ...,
       [-0.65382764],
       [-0.52109689],
       [ 0.54492668]])
```

1.5.4 Merging all the above features

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

In [46]:
```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(50000, 9)
(50000, 30)
(50000, 12211)
(50000, 1)
```

In [47]:
```
# 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 de
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_stand
X.shape
```

Out[47]:
```
(50000, 12251)
```

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

__ Computing Sentiment Scores__

In [49]:
```
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 th
for learning my students learn in many different ways using all of our senses
of techniques to help all my students succeed students in my class come from
for wonderful sharing of experiences and cultures including native americans
learners which can be seen through collaborative student project based learni
in my class love to work with hands on materials and have many different oppo
mastered having the social skills to work cooperatively with friends is a cru
montana is the perfect place to learn about agriculture and nutrition my stud
in the early childhood classroom i have had several kids ask me can we try co
and create common core cooking lessons where we learn important math and writ
food for snack time my students will have a grounded appreciation for the wor
of where the ingredients came from as well as how it is healthy for their bod
nutrition and agricultural cooking recipes by having us peel our own apples t
and mix up healthy plants from our classroom garden in the spring we will als
shared with families students will gain math and literature skills as well as
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
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975,
```

In [0]:
```
#Sentiment score for essay

sentiment_neg=[]
sentiment_neu=[]
sentiment_pos=[]
sentiment_comp=[]
```

In [51]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA

sid = SIA()
for sentence in tqdm(project_data['essay'].values):
    ss = sid.polarity_scores(sentence)
    sentiment_neg.append(ss['neg'])
    sentiment_neu.append(ss['neu'])
    sentiment_pos.append(ss['pos'])
    sentiment_comp.append(ss['compound'])
```

```
100%|██████████| 50000/50000 [01:30<00:00, 567.05it/s]
```

In [0]:
```
#Calculating number of words in the title

words_in_title = []
for i in project_data['project_title']:
    a = len(i.split())
    words_in_title.append(a)
```

In [0]:
```
#Combining all the essays. essay_3 and essay_4 contains NaN values only. So N
project_data['essay_text'] = project_data['project_essay_1'] + project_data['
```

In [0]:
```
#Calculating number of words in the essay_text(Combined essays)
words_in_essay = []
for i in project_data['essay_text']:
    a = len(i.split())
    words_in_essay.append(a)
```

In [0]:
```
#Adding the numerical features like sentiment score of essays, no. of words i

project_data['sentiment_neg'] = sentiment_neg
project_data['sentiment_pos'] = sentiment_pos
project_data['sentiment_comp'] = sentiment_comp
project_data['sentiment_neu'] = sentiment_neu

project_data['words_in_title'] = words_in_title
project_data['words_in_essay'] = words_in_essay
```

In [56]:
```
project_data.columns
```

Out[56]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_tit
le',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approve
d',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quanti
ty',
       'essay_text', 'sentiment_neg', 'sentiment_pos', 'sentiment_comp',
       'sentiment_neu', 'words_in_title', 'words_in_essay'],
      dtype='object')
```

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 [57]:

```
'''
import sys
import math

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

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

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

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

    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]))
  '''
```

Out[57]:
```
'\nimport sys\nimport math\n \nimport numpy as np\nfrom sklearn.model_selec
tion import GridSearchCV\nfrom sklearn.metrics import roc_auc_score\n\n# yo
u might need to install this one\nimport xgboost as xgb\n\nclass XGBoostCla
ssifier():\n    def __init__(self, num_boost_round=10, **params):\n
self.clf = None\n        self.num_boost_round = num_boost_round\n        se
lf.params = params\n        self.params.update({\'objective\': \'multi:soft
prob\'})\n \n    def fit(self, X, y, num_boost_round=None):\n        num_bo
ost_round = num_boost_round or self.num_boost_round\n        self.label2num
= {label: i for i, label in enumerate(sorted(set(y)))}\n        dtrain = xg
b.DMatrix(X, label=[self.label2num[label] for label in y])\n        self.cl
f = xgb.train(params=self.params, dtrain=dtrain, num_boost_round=num_boost_
round, verbose_eval=1)\n \n    def predict(self, X):\n        num2label =
{i: label for label, i in self.label2num.items()}\n        Y = self.predict
_proba(X)\n        y = np.argmax(Y, axis=1)\n        return np.array([num2l
abel[i] for i in y])\n \n    def predict_proba(self, X):\n        dtest = x
gb.DMatrix(X)\n        return self.clf.predict(dtest)\n \n    def score(sel
f, X, y):\n        Y = self.predict_proba(X)[:,1]\n        return roc_auc_s
core(y, Y)\n \n    def get_params(self, deep=True):\n        return self.pa
rams\n \n    def set_params(self, **params):\n        if \'num_boost_round
\' in params:\n            self.num_boost_round = params.pop(\'num_boost_ro
und\')\n        if \'objective\' in params:\n            del params[\'objec
tive\']\n        self.params.update(params)\n        return self\n    \n\nc
lf = XGBoostClassifier(eval_metric = \'auc\', num_class = 2, nthread = 4,)
\n\n#################################################################\n#
Change from here                                        #\n#####################
#############################################\nparameters = {\n    \'num_bo
ost_round\': [100, 250, 500],\n    \'eta\': [0.05, 0.1, 0.3],\n    \'max_de
pth\': [6, 9, 12],\n    \'subsample\': [0.9, 1.0],\n    \'colsample_bytree
\': [0.9, 1.0],\n}\n\nclf = GridSearchCV(clf, parameters)\nX = np.array
([[1,2], [3,4], [2,1], [4,3], [1,0], [4,5]])\nY = np.array([0, 1, 0, 1, 0,
1])\nclf.fit(X, Y)\n\n# print(clf.grid_scores_)\nbest_parameters, score, _
= max(clf.grid_scores_, key=lambda x: x[1])\nprint(\'score:\', score)\nfor
param_name in sorted(best_parameters.keys()):\n    print("%s: %r" % (param_
name, best_parameters[param_name]))\n  '
```

Splitting the data into train and test

In [0]:
```
y = project_data['project_is_approved'].values
x = project_data.drop(['project_is_approved'], axis=1)
```

In [0]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.33, str
```

vectorizing the categorical features

In [60]:
```
#school_state
vectorizer = CountVectorizer()
vectorizer.fit(x_train['school_state'].values)

x_train_state = vectorizer.transform(x_train['school_state'].values)
x_test_state = vectorizer.transform(x_test['school_state'].values)

print("After Vectorizations:")
print(x_train_state.shape, y_train.shape)
print(x_test_state.shape, y_test.shape)
```

```
After Vectorizations:
(33500, 51) (33500,)
(16500, 51) (16500,)
```

In [61]:
```
#teacher_prefix

vectorizer = CountVectorizer()
vectorizer.fit(x_train['teacher_prefix'].values.astype('U'))

x_train_teacher = vectorizer.transform(x_train['teacher_prefix'].values.astyp
x_test_teacher = vectorizer.transform(x_test['teacher_prefix'].values.astype(

print("After Vectorizations:")
print(x_train_teacher.shape, y_train.shape)
print(x_test_teacher.shape, y_test.shape)
```

```
After Vectorizations:
(33500, 6) (33500,)
(16500, 6) (16500,)
```

In [62]:
```
#Project_grade_category

vectorizer = CountVectorizer()
vectorizer.fit(x_train['project_grade_category'].values)

x_train_grade = vectorizer.transform(x_train['project_grade_category'].values
x_test_grade = vectorizer.transform(x_test['project_grade_category'].values)

print("After Vectorizations:")
print(x_train_grade.shape, y_train.shape)
print(x_test_grade.shape, y_test.shape)
```

```
After Vectorizations:
(33500, 4) (33500,)
(16500, 4) (16500,)
```

In [63]:
```
#project_subject_categories


vectorizer = CountVectorizer()
vectorizer.fit(x_train['clean_categories'].values)

x_train_categories = vectorizer.transform(x_train['clean_categories'].values)
x_test_categories = vectorizer.transform(x_test['clean_categories'].values)

print("After Vectorizations:")
print(x_train_categories.shape, y_train.shape)
print(x_test_categories.shape, y_test.shape)
```

```
After Vectorizations:
(33500, 9) (33500,)
(16500, 9) (16500,)
```

In [64]:
```
#project_subject_subcategories


vectorizer = CountVectorizer()
vectorizer.fit(x_train['clean_subcategories'].values)

x_train_subcategories = vectorizer.transform(x_train['clean_subcategories'].v
x_test_subcategories = vectorizer.transform(x_test['clean_subcategories'].val

print("After Vectorizations:")
print(x_train_subcategories.shape, y_train.shape)
print(x_test_subcategories.shape, y_test.shape)
```

```
After Vectorizations:
(33500, 30) (33500,)
(16500, 30) (16500,)
```

Normalizing the Numerical features

In [65]:
```
#Price
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
normalizer.fit(x_train['price'].values.reshape(1, -1))

x_train_price = normalizer.transform(x_train['price'].values.reshape(1, -1))
x_test_price = normalizer.transform(x_test['price'].values.reshape(1, -1))

print('AFter vectorizations:')
print(x_train_price.shape, y_train.shape)
print(x_test_price.shape, y_test.shape)
```

```
AFter vectorizations:
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [66]:
```
#Teacher_number_of_previously_posted_projects

normalizer.fit(x_train['teacher_number_of_previously_posted_projects'].values

x_train_previous_projects = normalizer.transform(x_train['teacher_number_of_p
x_test_previous_projects = normalizer.transform(x_test['teacher_number_of_pre

print('After Vectorizations:')
print(x_train_previous_projects.shape, y_train.shape)
print(x_test_previous_projects.shape, y_test.shape)
```

```
After Vectorizations:
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [67]:
```
#Quantity

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

x_train_quantity = normalizer.transform(x_train['quantity'].values.reshape(1,
x_test_quantity = normalizer.transform(x_test['quantity'].values.reshape(1, -

print('After Vectorizations:')
print(x_train_quantity.shape, y_train.shape)
print(x_test_quantity.shape, y_test.shape)
```

```
After Vectorizations:
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [68]:
```
#Sentiment_neg

normalizer.fit(x_train['sentiment_neg'].values.reshape(1, -1))

x_train_sentiment_neg = normalizer.transform(x_train['sentiment_neg'].values.
x_test_sentiment_neg = normalizer.transform(x_test['sentiment_neg'].values.re

print('After Vectorizations:')
print(x_train_sentiment_neg.shape, y_train.shape)
print(x_test_sentiment_neg.shape, y_test.shape)
```

```
After Vectorizations:
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [69]:
```
#Sentiment_pos

normalizer.fit(x_train['sentiment_pos'].values.reshape(1, -1))

x_train_sentiment_pos = normalizer.transform(x_train['sentiment_pos'].values.
x_test_sentiment_pos = normalizer.transform(x_test['sentiment_pos'].values.re

print('After Vectorizations:')
print(x_train_sentiment_pos.shape, y_train.shape)
print(x_test_sentiment_pos.shape, y_test.shape)
```

```
After Vectorizations:
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [70]:
```
#Sentiment_comp

normalizer.fit(x_train['sentiment_comp'].values.reshape(1, -1))

x_train_sentiment_comp = normalizer.transform(x_train['sentiment_comp'].value
x_test_sentiment_comp = normalizer.transform(x_test['sentiment_comp'].values.

print('After Vectorizations:')
print(x_train_sentiment_comp.shape, y_train.shape)
print(x_test_sentiment_comp.shape, y_test.shape)
```

```
After Vectorizations:
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [71]:
```
#Sentiment_neu

normalizer.fit(x_train['sentiment_neu'].values.reshape(1, -1))

x_train_sentiment_neu = normalizer.transform(x_train['sentiment_neu'].values.
x_test_sentiment_neu = normalizer.transform(x_test['sentiment_neu'].values.re

print('After Vectorizations:')
print(x_train_sentiment_neu.shape, y_train.shape)
print(x_test_sentiment_neu.shape, y_test.shape)
```

```
After Vectorizations:
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [72]:
```
#words_in_title

normalizer.fit(x_train['words_in_title'].values.reshape(1, -1))

x_train_words_in_title = normalizer.transform(x_train['words_in_title'].value
x_test_words_in_title = normalizer.transform(x_test['words_in_title'].values.

print('After Vectorizations:')
print(x_train_words_in_title.shape, y_train.shape)
print(x_test_words_in_title.shape, y_test.shape)
```

```
After Vectorizations:
(1, 33500) (33500,)
(1, 16500) (16500,)
```

In [73]:
```
#words_in_essay

normalizer.fit(x_train['words_in_essay'].values.reshape(1, -1))

x_train_words_in_essay = normalizer.transform(x_train['words_in_essay'].value
x_test_words_in_essay = normalizer.transform(x_test['words_in_essay'].values.

print('After Vectorizations:')
print(x_train_words_in_essay.shape, y_train.shape)
print(x_test_words_in_essay.shape, y_test.shape)
```

```
After Vectorizations:
(1, 33500) (33500,)
(1, 16500) (16500,)
```

2. TruncatedSVD

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

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

In [0]:
```
#Combining Essay, combined essays and title columns

project_data['combined_text'] = project_data['project_title'] + project_data[
```

In [76]:
```
#Getting idf values
#Reference https://stackoverflow.com/questions/48431173/is-there-a-way-to-get

tf = TfidfVectorizer(use_idf=True)
tf.fit_transform(project_data['combined_text'])

idf = tf.idf_
idf
```

Out[76]:
```
array([ 7.30893878,  5.93092033, 11.1266511 , ..., 11.1266511 ,
       11.1266511 , 10.02803881])
```

In [77]:
```
feature_array = np.array(tf.get_feature_names())
feature_array.shape
```

Out[77]:
```
(66232,)
```

In [78]:
```
a = np.argsort(idf)
a = a[:2000]
a.shape
```

Out[78]:
```
(2000,)
```

In [79]:
```
top_words = np.take(feature_array, a)
top_words.shape
```

Out[79]:
```
(2000,)
```

In [0]:
```
top_words = top_words.tolist()
```

In [0]:
```
corpus = project_data['essay']
```

In [82]:
```
co_matrix = np.zeros((2000, 2000))
co_matrix.shape
```

Out[82]:
```
(2000, 2000)
```

2.2 Computing Co-occurance matrix

In [0]:
```
# please write all the code with proper documentation, and proper titles for
# 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 debu
# make sure you featurize train and test data separatly

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

In [0]:
```
#co-occurence matrix
context_window = 5
```

In [87]:
```
for sent in tqdm(corpus): #Obtaining each sentence
    words = sent.split() #Obtaining each word of a sent
    for index, word in enumerate(words): #obtaining the index of each word fr
        if word in top_words: #checking whether the word is in top_words
            for i in range(max(index - context_window, 0), min(index + contex
                if words[i] in top_words:
                    co_matrix[top_words.index(words[i]), top_words.index(word
                else:
                    continue
        else:
            continue
```

```
100%|██████████| 50000/50000 [20:06<00:00, 45.61it/s]
```

In [88]:
```
np.fill_diagonal(co_matrix, 0)
co_matrix
```

Out[88]:
```
array([[ 0., 18., 62., ...,  1.,  1.,  1.],
       [18.,  0., 50., ...,  2.,  1.,  0.],
       [62., 50.,  0., ..., 23., 47., 71.],
       ...,
       [ 1.,  2., 23., ...,  0.,  0.,  0.],
       [ 1.,  1., 47., ...,  0.,  0.,  3.],
       [ 1.,  0., 71., ...,  0.,  3.,  0.]])
```

In [1]:
```
###Sample co-occurence matrix
```

In [90]:
```
import numpy as np

corpus = ['abc def ijk pqr', 'pqr klm opq', 'lmn pqr xyz abc def pqr abc']

top_words = ['abc', 'pqr', 'def']

context_window = 2
cooccurence_matrix = np.zeros((3, 3))
cooccurence_matrix.shape
```

Out[90]: (3, 3)

In [91]:
```
for sent in tqdm(corpus):
    words = sent.split()
    for index, word in enumerate(words):
        if word in top_words:
            for i in range(max(index - context_window, 0), min(index + contex
                if words[i] in top_words:
                    cooccurence_matrix[top.index(words[i]), top_words.index(w
                else:
                    continue
        else:
            continue
```

100%|██████████| 3/3 [00:00<00:00, 6429.69it/s]

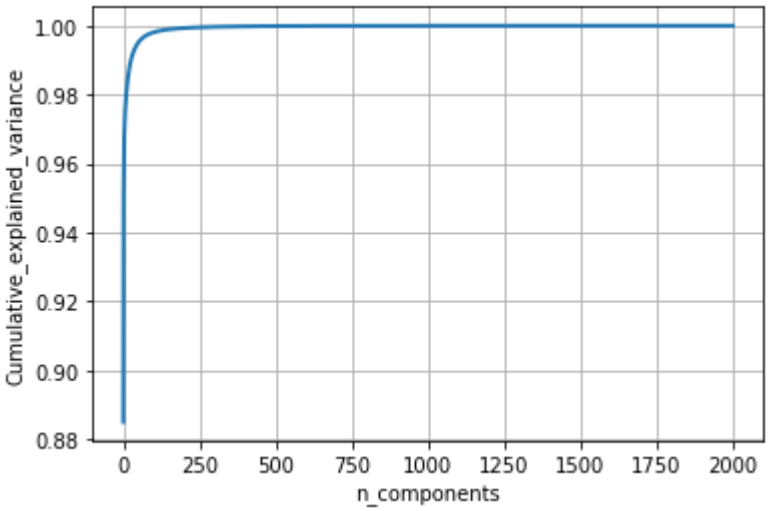
In [92]:
```
np.fill_diagonal(cooccurence_matrix, 0)
cooccurence_matrix
```

Out[92]:
```
array([[0., 3., 3.],
       [3., 0., 2.],
       [3., 2., 0.]])
```

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

In [0]:
```
# please write all the code with proper documentation, and proper titles for
# 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 debu
# make sure you featurize train and test data separatly

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

In [0]:
```
from sklearn import decomposition
pca = decomposition.PCA()


#For Train
#x_train_essay_tfidf = x_train_essay_tfidf.toarray()

pca.n_components = 2000
pca_data = pca.fit_transform(co_matrix)

percentage_var_explained = pca.explained_variance_ / np.sum(pca.explained_var

cum_var_explained = np.cumsum(percentage_var_explained)

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

plt.clf()
plt.plot(cum_var_explained, linewidth=2)
plt.axis('tight')
plt.grid()
plt.xlabel('n_components')
plt.ylabel('Cumulative_explained_variance')
plt.show()
```



In [0]:
```
#Most of the data can be preserved with only around 200 components

from sklearn.decomposition import TruncatedSVD
svd = TruncatedSVD(n_components=200, n_iter=100)

co_matrix = svd.fit_transform(co_matrix)
print(co_matrix.shape)
```

```
(2000, 200)
```

In [0]:
```
dic = {}
dic = dic.fromkeys(top_words, 0)
```

In [0]:
```
for key, value in dic.items():
    dic[key] = list(co_matrix[:, 0])  #getting a row from com_matrix and addi
```

Vectorizing Title

In [0]:
```
x_train_title = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(x_train['project_title'].values): # for each review/sent
    vector = np.zeros(2000) # 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 top_words:

            vector += dic[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    x_train_title.append(vector)

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

```
100%|██████████| 33500/33500 [00:17<00:00, 1917.72it/s]

33500
2000
```

In [0]:
```
x_test_title = []; # the avg-w2v for each sentence/review is stored in this l
for sentence in tqdm(x_test['project_title'].values): # for each review/sente
    vector = np.zeros(2000) # 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 top_words:

            vector += dic[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    x_test_title.append(vector)

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

```
100%|██████████| 16500/16500 [00:08<00:00, 1893.79it/s]

16500
2000
```

Vectorizing Essay

In [0]:
```
x_train_essay = []; # the avg-w2v for each sentence/review is stored in this
for sentence in tqdm(x_train['essay'].values): # for each review/sentence
    vector = np.zeros(2000) # 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 top_words:

            vector += dic[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    x_train_essay.append(vector)

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

```
100%|██████████| 33500/33500 [12:29<00:00, 44.68it/s]

33500
2000
```

In [0]:
```
x_test_essay = []; # the avg-w2v for each sentence/review is stored in this l
for sentence in tqdm(x_test['essay'].values): # for each review/sentence
    vector = np.zeros(2000) # 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 top_words:

            vector += dic[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    x_test_essay.append(vector)

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

```
100%|██████████| 16500/16500 [06:08<00:00, 44.81it/s]

16500
2000
```

2.4 Merge the features from step 3 and step 4

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

In [0]:
```
#Reshaping the numerical features

x_train_price = x_train_price.reshape(-1, 1)
x_test_price = x_test_price.reshape(-1, 1)

x_train_previous_projects = x_train_previous_projects.reshape(-1, 1)
x_test_previous_projects = x_test_previous_projects.reshape(-1, 1)

x_train_quantity = x_train_quantity.reshape(-1, 1)
x_test_quantity = x_test_quantity.reshape(-1, 1)

x_train_sentiment_neg = x_train_sentiment_neg.reshape(-1, 1)
x_test_sentiment_neg = x_test_sentiment_neg.reshape(-1, 1)

x_train_sentiment_pos = x_train_sentiment_pos.reshape(-1, 1)
x_test_sentiment_pos = x_test_sentiment_pos.reshape(-1, 1)

x_train_sentiment_comp = x_train_sentiment_comp.reshape(-1, 1)
x_test_sentiment_comp = x_test_sentiment_comp.reshape(-1, 1)

x_train_sentiment_neu = x_train_sentiment_neu.reshape(-1, 1)
x_test_sentiment_neu = x_test_sentiment_neu.reshape(-1, 1)

x_train_words_in_title = x_train_words_in_title.reshape(-1, 1)
x_test_words_in_title = x_test_words_in_title.reshape(-1, 1)

x_train_words_in_essay = x_train_words_in_essay.reshape(-1, 1)
x_test_words_in_essay = x_test_words_in_essay.reshape(-1, 1)
```

In [0]:
```
from scipy.sparse import hstack

x_train = hstack((x_train_price, x_train_previous_projects, x_train_quantity,
                  x_train_sentiment_neu, x_train_words_in_title, x_train_word
                  x_train_subcategories, x_train_title, x_train_title))

x_test = hstack((x_test_price, x_test_previous_projects, x_test_quantity, x_t
                 x_test_words_in_title, x_test_words_in_essay, x_test_state,
                 x_test_essay))
```

2.5 Apply XGBoost on the Final Features from the above section

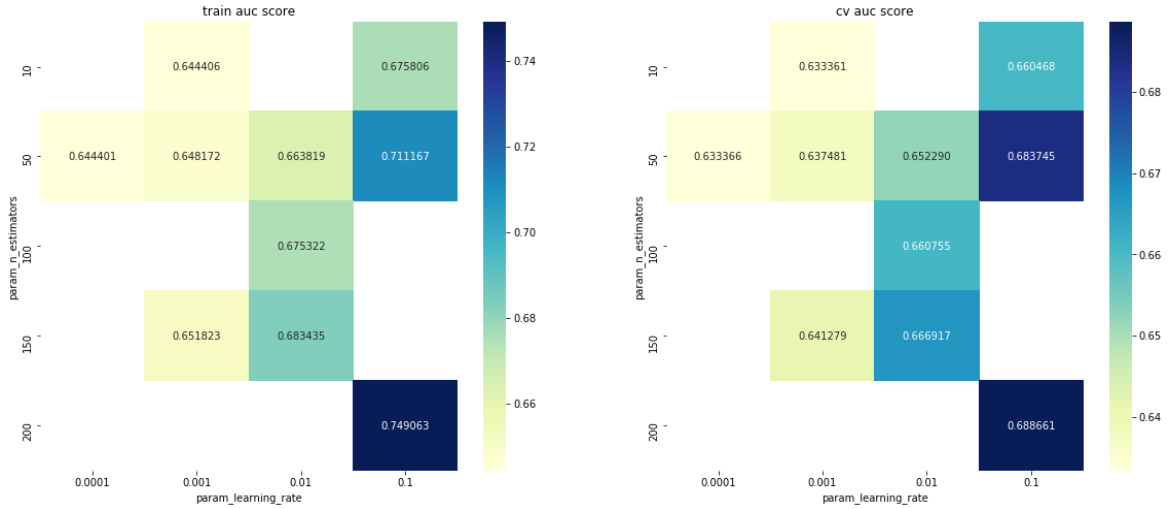
https://xgboost.readthedocs.io/en/latest/python/python_intro.html (https://xgboost.readthedocs.io/en/latest/python/python_intro.html)

In [0]:
```
# No need to split the data into train and test(cv)
# use the Dmatrix and apply xgboost on the whole data
# please check the Quora case study notebook as reference

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

In [0]:
```
import xgboost as xgb

#Hyperparameter Tuning
from sklearn.model_selection import RandomizedSearchCV

tuned_parameters = {'n_estimators':[10, 50, 100, 150, 200], 'learning_rate':[
xgb_model = xgb.XGBClassifier()
clf = RandomizedSearchCV(xgb_model, tuned_parameters, cv=5, scoring='roc_auc'
clf.fit(x_train, y_train)
```

Out[101]:
```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
                   estimator=XGBClassifier(base_score=0.5, booster='gbtre
e',
                                           colsample_bylevel=1,
                                           colsample_bynode=1,
                                           colsample_bytree=1, gamma=0,
                                           learning_rate=0.1, max_delta_ste
p=0,
                                           max_depth=3, min_child_weight=1,
                                           missing=None, n_estimators=100,
                                           n_jobs=1, nthread=None,
                                           objective='binary:logistic',
                                           random_state=0, reg_alpha=0,
                                           reg_lambda=1, scale_pos_weight=
1,
                                           seed=None, silent=None, subsampl
e=1,
                                           verbosity=1),
                   iid='warn', n_iter=10, n_jobs=None,
                   param_distributions={'learning_rate': [0.0001, 0.001, 0.
01,
                                                          0.1],
                                        'n_estimators': [10, 50, 100, 150,
                                                         200]},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=True, scoring='roc_auc', verbose=0)
```

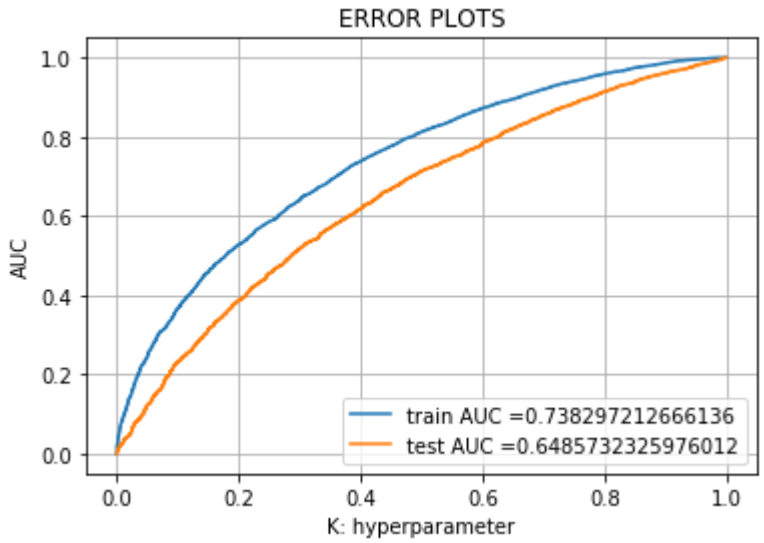
In [0]:
```
results = pd.DataFrame.from_dict(clf.cv_results_)

#plotting heatmap of train and cv auc score
plt.figure(figsize=(20,8))
plt.subplot(1,2,1)
train_pvt = pd.pivot_table(pd.DataFrame(clf.cv_results_),values='mean_train_s
ax = sns.heatmap(train_pvt,annot=True,cmap="YlGnBu",fmt="f")
plt.title("train auc score")
plt.subplot(1,2,2)
test_pvt=pvt = pd.pivot_table(pd.DataFrame(clf.cv_results_),values='mean_test
ax1 = sns.heatmap(test_pvt,annot=True,cmap="YlGnBu",fmt="f")
plt.title("cv auc score")
plt.show()
```



In [0]:
```
clf.best_params_
```

Out[103]:
```
{'learning_rate': 0.1, 'n_estimators': 200}
```

In [0]:
```
from sklearn.metrics import roc_curve, auc

xgb_model = xgb.XGBClassifier(learning_rate=0.1, n_estimators=250)
xgb_model.fit(x_train, y_train)

y_train_pred = clf.predict_proba(x_train)[:, 1]
y_test_pred = clf.predict_proba(x_test)[:, 1]

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.grid()
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_t
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")

plt.show()
```



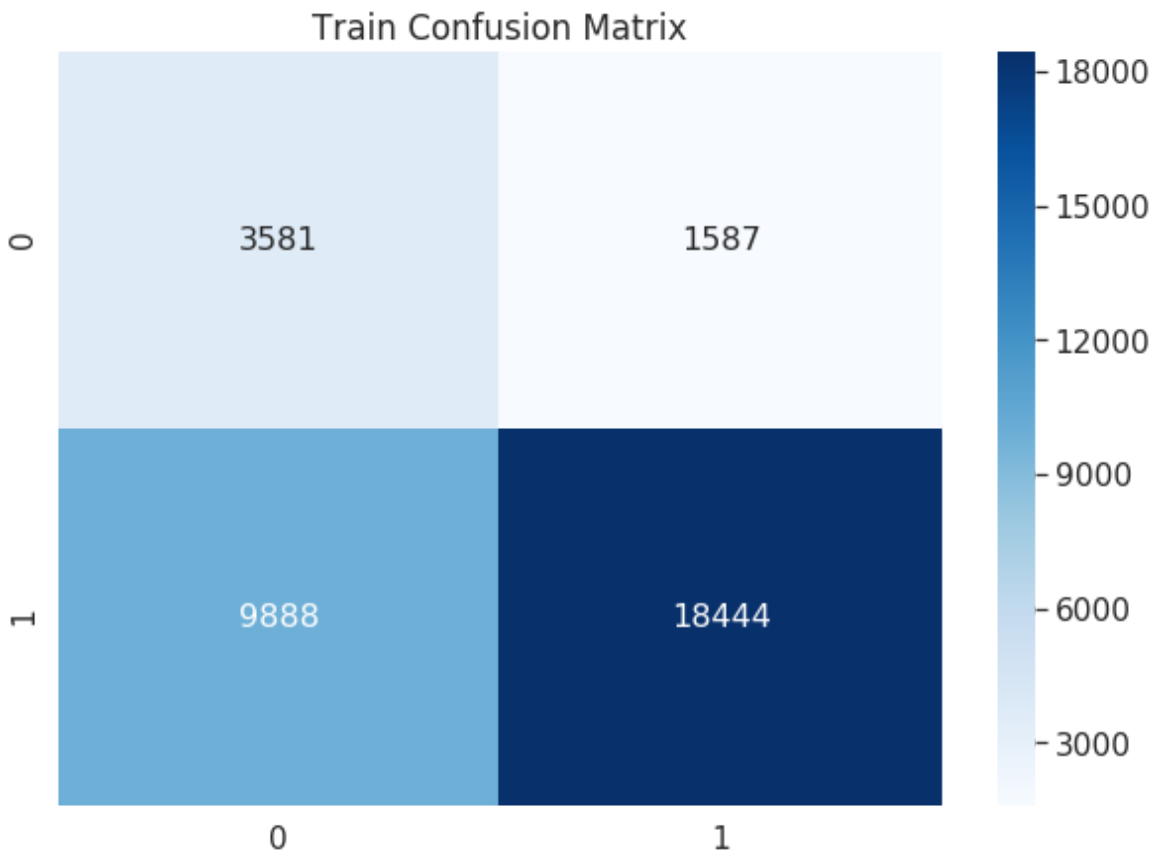
In [0]:
```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very h
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for thresho
    return t

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

In [0]:
```
#Confusion matrix

from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred, best_t)))
```

```
the maximum value of tpr*(1-fpr) 0.45108636144887976 for threshold 0.845
Train confusion matrix
[[ 3581  1587]
 [ 9888 18444]]
Test confusion matrix
[[1841  705]
 [7178 6776]]
```

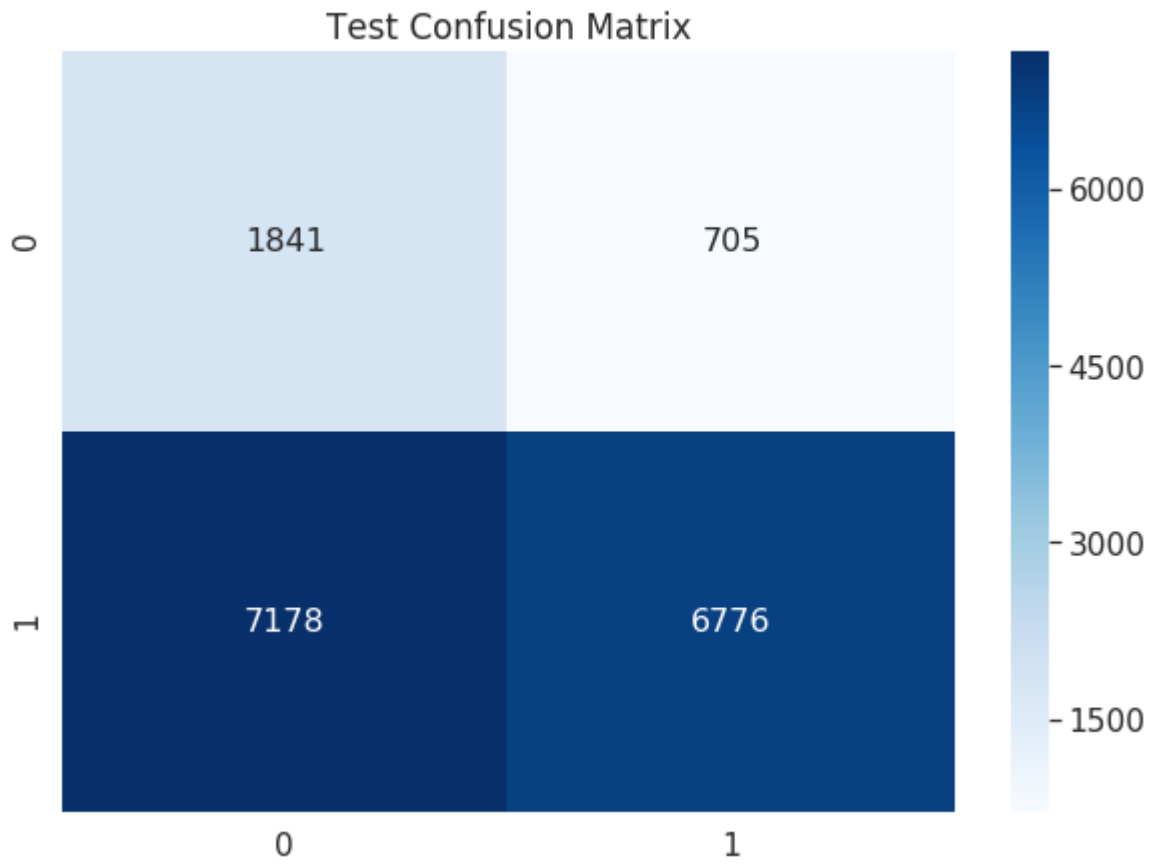
In [0]:
```
#Train Confusion matrix

#Reference- https://www.kaggle.com/agungor2/various-confusion-matrix-plots

y_train_predicted= predict_with_best_t(y_train_pred, best_t)

df_cm = pd.DataFrame(confusion_matrix(y_train,y_train_predicted), columns=np.

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Train Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[107]: <matplotlib.axes._subplots.AxesSubplot at 0x7febf98f6b00>



In [0]:
```
#Test Confusion matrix
y_test_predicted=predict_with_best_t(y_test_pred, best_t)
df_cm = pd.DataFrame(confusion_matrix(y_test,y_test_predicted ), columns=np.u

plt.figure(figsize = (10,7))
sns.set(font_scale=1.4)#for label size
plt.title('Test Confusion Matrix')
sns.heatmap(df_cm, cmap="Blues", annot=True, fmt='g', annot_kws={"size": 16})
```

Out[108]: <matplotlib.axes._subplots.AxesSubplot at 0x7febf9903518>



3. Conclusion

In [5]:

```
# Please write down few lines about what you observed from this assignment.
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer", "Model", "Hyper Parameter", "Test AUC"]
x.add_row(["AVG W2V", "XGBoost", "Max Depth:3 , n_estimators:100", 0.648])
print(x)
```

```
+------------+---------+--------------------------------+----------+
| Vectorizer |  Model  |        Hyper Parameter         | Test AUC |
+------------+---------+--------------------------------+----------+
|  AVG W2V   | XGBoost | Max Depth:3 , n_estimators:100 |  0.648   |
+------------+---------+--------------------------------+----------+
```