DonorsChoose

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

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

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

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

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`project_essay_4`	Fourth application essay
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

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

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

In [1]:

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

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
```

```
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
warnings.filterwarnings("ignore")
```

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

1.1 Reading Data

In [2]:

```
project_data = pd.read_csv('E:\\Machine Learning\\Dataset\\train_data.csv').sample(50000)
resource_data = pd.read_csv('E:\\Machine Learning\\Dataset\\resources.csv')
```

In [3]:

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

```
Number of data points in train data (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 [4]:

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


#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


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

```
project_data.head(2)
project_data.project_is_approved.value_counts()
```

Out[4]:

```
1    42427
0     7573
Name: project_is_approved, dtype: int64
```

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 [6]:

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

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

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, 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 [7]:

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

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

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

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

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

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

1.4 Text preprocessing

In [8]:

```
# 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	Date	project_grade_category	project_t
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5	Sens Tools Fo
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Grades PreK-2	Mc Learn wi Mc Lister Ce

In [10]:

```
# printing some random reviews
```

```
# 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)
```

```
Imagine being 8-9 years old. You're in your third grade classroom. You see bright lights, the kid
next to you is chewing gum, the birds are making noise, the street outside is buzzing with cars, i
t's hot, and your teacher is asking you to focus on learning. Ack! You need a break! So do my stud
ents.Most of my students have autism, anxiety, another disability, or all of the above. It is toug
h to focus in school due to sensory overload or emotions. My students have a lot to deal with in s
chool, but I think that makes them the most incredible kids on the planet. They are kind, caring,
and sympathetic. They know what it's like to be overwhelmed, so they understand when someone else
is struggling. They are open-minded and compassionate. They are the kids who will someday change t
he world.It is tough to do more than one thing at a time. When sensory overload gets in the way, i
t is the hardest thing in the world to focus on learning. My students need many breaks throughout
the day, and one of the best items we've used is a Boogie Board. If we had a few in our own classr
oom, my students could take a break exactly when they need one, regardless of which other rooms in
the school are occupied. Many of my students need to do something with their hands in order to
focus on the task at hand. Putty will give the sensory input they need in order to focus, it will
calm them when they are overloaded, it will help improve motor skills, and it will make school mor
e fun.When my students are able to calm themselves down, they are ready to learn. When they are ab
le to focus, they will learn more and retain more. They will get the sensory input they need and i
t will prevent meltdowns (which are scary for everyone in the room). This will lead to a better, h
appier classroom community that is able to learn the most they can in the best way possible.
==================================================
My students love working on the computer. This year we have been blessed with many online
resources. Most of them use videos/sound in at least part of teaching. These resources are great f
or review and I would like them to get the maximum benefit from them.We are a very small school in
a very high poverty area. I have great students who are trying to succeed in life. My students are
not able to bring in their own headphones as they do at other schools. We are a tight knit school
and we look out for each other and assist in any way we can.There are a plethora of resources avai
lable to use on the internet. They are great for reviewing previously taught concepts. A lot of th
ese resources use sound in their program. If the students are not able to hear the program they mi
ss out on a lot, but we can't have 20 computers making noise.My students work hard, but still stru
ggle with basic concepts. Online resources are great since they provide learning/review in a fun f
ormat - usually it's a game which is completely engaging to the students. It's a great way to \"tr
ick\" them into learning!!
==================================================
Our school is a Title I school with a population of high poverty and many at-risk students.    Tea
chers at our school have high expectations for these students regardless of their economic and edu
cational background.  They are eager to learn and are ready to meet their educational goals.The st
udents in my classroom enjoy learning.  They especially get pleasure from doing thematic units and
usually do the activities that go above and beyond what is expected of them.   Technology is in
incorporated into the learning targets when possible to help students be successful in their meeti
ng the second grade standards.This year my students were fortunate to move into a brand new school
because the old school that included several portables was literally falling apart.  It was exciti
ng seeing the students' faces and they toured the new school.  Each classroom has an LCD projector
, a document camera, three PC computers and a few First Generation iPads.   \r\n\r\nThe students i
n my classroom are like most digital students that enjoy the usage of technology in the classroom.
They love to read electronic books, do research in the computer and like to type reports in the co
mputer.  Having students use computer programs that build their computer skills are essential in p
reparing students for the 21st century learning.  Students are able to connect with their world an
d this enables them learn from others and to share their own ideas. I also use technology often to
differentiate the curriculum so that students are learning on their own developmental levels and t
hus are able to pursue their unique interestsThis project will help students integrate language ar
ts, science and social studies across the curriculum. Narratives like personal stories, drama, sci
ence and social studies, will be used in audio or photo story presentations.   The utilization of
multimedia software, including image, video and audio editing will help students have a deeper lev
el of understanding of technology by creating podcasts that can be shared and heard by many
persons.
==================================================
```

In [11]:

```
# 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 [12]:

```
sent = decontracted(project_data['essay'].values[2000])
print(sent)
print("="*50)
```

```
As a teacher in a Title One school , my students are faced with several challenges both in and out
of the classroom. Despite the many challenges they face they come ready to learn. My goal is to cr
eate a classroom that enables them to have a future of their choice. \r\n For most of my students
English is a second language. 75% of our students are on free or reduced lunches. They are childre
n who are excited about learning and want a better life than what their parents have, but they nee
d help in reaching their goal. With the budget issues in the state of North Carolina, we do not ha
ve the money for the technology my students will need to help them succeed in the future. I want m
y students to change the world. And to do that, they need access to the same tools as other
students. They need access to technology for research, learning and collaborating on higher level
thinking task. All of which they can get done on these iPad Minis.\r\nWith the use of the iPads in
the classroom, my students will be able to access more than 200,000 apps to help them supplement t
heir math skills such as addition and subtraction facts. They will be able to use these apps to he
lp with their reading and writing of the English language. They will be able to complete
collaborate writing projects. Form partnerships with other 2nd graders around the world and have u
nlimited access to online books.Our limited resources will not allow us as a school to purchase it
ems such as iPads.How can we expect our students to compete with the rest of the nation or the wor
ld if the resources are not there. Not only will the technology encourage these students to learn,
but will assist in them making a difference in the world that will soon become theirs.They need ac
cess to technology for research, learning and collaborating on higher level thinking task. All of
which they can get done on these iPad Minis\r\nnannan
==================================================
```

In [13]:

```
# \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)
```

```
As a teacher in a Title One school , my students are faced with several challenges both in and out
of the classroom. Despite the many challenges they face they come ready to learn. My goal is to cr
eate a classroom that enables them to have a future of their choice.    For most of my students  E
nglish is a second language. 75% of our students are on free or reduced lunches. They are children
who are excited about learning and want a better life than what their parents have, but they need
help in reaching their goal. With the budget issues in the state of North Carolina, we do not have
the money for the technology my students will need to help them succeed in the future. I want my s
tudents to change the world. And to do that, they need access to the same tools as other students.
They need access to technology for research, learning and collaborating on higher level thinking t
ask. All of which they can get done on these iPad Minis.  With the use of the iPads in the classro
om, my students will be able to access more than 200,000 apps to help them supplement their math s
kills such as addition and subtraction facts. They will be able to use these apps to help with the
ir reading and writing of the English language. They will be able to complete collaborate writing
projects. Form partnerships with other 2nd graders around the world and have unlimited access to o
nline books.Our limited resources will not allow us as a school to purchase items such as
iPads.How can we expect our students to compete with the rest of the nation or the world if the re
sources are not there. Not only will the technology encourage these students to learn, but will as
sist in them making a difference in the world that will soon become theirs.They need access to tec
hnology for research, learning and collaborating on higher level thinking task. All of which they
can get done on these iPad Minis  nannan
```

In [14]:

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

```
As a teacher in a Title One school my students are faced with several challenges both in and out o
f the classroom Despite the many challenges they face they come ready to learn My goal is to
create a classroom that enables them to have a future of their choice For most of my students Engl
ish is a second language 75 of our students are on free or reduced lunches They are children who a
re excited about learning and want a better life than what their parents have but they need help i
n reaching their goal With the budget issues in the state of North Carolina we do not have the mon
ey for the technology my students will need to help them succeed in the future I want my students
to change the world And to do that they need access to the same tools as other students They need
access to technology for research learning and collaborating on higher level thinking task All of
which they can get done on these iPad Minis With the use of the iPads in the classroom my students
will be able to access more than 200 000 apps to help them supplement their math skills such as ad
dition and subtraction facts They will be able to use these apps to help with their reading and wr
iting of the English language They will be able to complete collaborate writing projects Form part
nerships with other 2nd graders around the world and have unlimited access to online books Our lim
ited resources will not allow us as a school to purchase items such as iPads How can we expect our
students to compete with the rest of the nation or the world if the resources are not there Not on
ly will the technology encourage these students to learn but will assist in them making a
difference in the world that will soon become theirs They need access to technology for research l
earning and collaborating on higher level thinking task All of which they can get done on these iP
ad Minis nannan
```

In [15]:

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

In [16]:

```
# 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.lower() not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:40<00:00, 1233.15it/s]
```

In [17]:

```
# after preprocesing
print(preprocessed_essays[2000])
project_data['preprocessed_essays']=preprocessed_essays
```

```
teacher title one school students faced several challenges classroom despite many challenges face
come ready learn goal create classroom enables future choice students english second language 75 s
tudents free reduced lunches children excited learning want better life parents need help reaching
goal budget issues state north carolina not money technology students need help succeed future wan
t students change world need access tools students need access technology research learning collab
orating higher level thinking task get done ipad minis use ipads classroom students able access 20
0 000 apps help supplement math skills addition subtraction facts able use apps help reading writi
ng english language able complete collaborate writing projects form partnerships 2nd graders aroun
d world unlimited access online books limited resources not allow us school purchase items ipads e
xpect students compete rest nation world resources not not technology encourage students learn ass
ist making difference world soon become need access technology research learning collaborating hig
her level thinking task get done ipad minis nannan
```

1.4.1 Converting Essay to Number of Words

In [18]:

```
project_data['totalwords_essay'] = project_data['preprocessed_essays'].str.split().str.len()
```

1.5 Preprocessing of `project_title`

In [19]:

```
# Combining all the above statemennts
#from tqdm import tqdm
preprocessed_project_title = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = 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_project_title.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:01<00:00, 29380.47it/s]
```

In [20]:

```
print(preprocessed_project_title[2000])
print("="*50)
project_data['preprocessed_project_title']=preprocessed_project_title
project_data.head(5)
```

```
ipads inquiry
==================================================
```

Out[20]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_
76127	37728	p043609	3f60494c61921b3b43ab61bdde2904df	Ms.	UT	2016-04-27 00:31:25	Grades 3-5	Sen Tool Fo
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Grades PreK-2	M Lear w M Liste

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category	project_
79026	139722	p182545	22460c54072bd0cf958cc8349fac8b8f	Ms.	CA	2016-04-27 02:02:27	Grades 3-5	21st Cer Lear Multim
23374	72317	p087808	598621c141cda5fb184ee7e8ccdd3fcc	Ms.	CA	2016-04-27 02:04:15	Grades PreK-2	iPa Lear
70898	128817	p239087	11a60ddd63717c59fdd5a13ea92d34aa	Mrs.	KY	2016-04-27 08:02:22	Grades 3-5	Tim Kids.. Learn A Science m

5 rows × 21 columns

1.5.1 Converting Title to Number of Words

In [21]:

```
project_data['totalwords_title'] =
project_data['preprocessed_project_title'].str.split().str.len()
```

1.6 Preprocessing Grades

In [22]:

```
#Preprocessing grades i.e removing all the spaces from the grades and replace - by _



pre_grades = []

for grade in tqdm(project_data['project_grade_category'].values):
    sent = re.sub('[^A-Za-z0-9]+', '_', grade)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    pre_grades.append(sent.lower().strip())


project_data['grade_category']=pre_grades
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 50000/50000
[00:00<00:00, 114526.12it/s]
```

1.7 Preparing data for models

In [23]:

```
project_data.columns
```

Out[23]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay',
       'preprocessed_essays', 'totalwords_essay', 'preprocessed_project_title',
       'totalwords_title', 'grade_category'],
      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
```

1.10 Combining project_data & resources

In [24]:

```
#Combining the data from the resources from the project data and resoure file for quantity and pri
ce
project_data['combined']=new_data_train=project_data['preprocessed_essays']+project_data['preproces
sed_project_title']
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 [25]:

```
#replacing all the nan values from the teacher prefix to blank_space

project_data.teacher_prefix=project_data.teacher_prefix.fillna('')
```

In [26]:

```
#Seprating the values of approved projects from the whole data i.e removing the target value from
the data
X=project_data
```

In [27]:

```
y =X['project_is_approved'].values

X.head(5)
y.shape
```

Out[27]:

```
(50000,)
```

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_` values
- step 2 Compute the co-occurance matrix with these 2k words, with window size=5 (ref)
- step 3 Use TruncatedSVD on calculated co-occurance matrix and reduce its dimensions, choose the number of components (`n_components`) using elbow method

> - 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**
- **step 6:Hyper parameter tuning (Consider any two hyper parameters)**
  - **Find the best hyper parameter which will give the maximum AUC value**
  - **Find the best hyper paramter using k-fold cross validation or simple cross validation data**
  - **Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning**

2.2 Vectorizing Numericals Features

2.1 Spliting of data

In [28]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.25, stratify=y_train)


print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)
print(X_train.columns)
```

```
(28125, 26) (28125,)
(9375, 26) (9375,)
(12500, 26) (12500,)
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'Date', 'project_grade_category', 'project_title', 'project_essay_1',
       'project_essay_2', 'project_essay_3', 'project_essay_4',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay',
       'preprocessed_essays', 'totalwords_essay', 'preprocessed_project_title',
       'totalwords_title', 'grade_category', 'combined', 'price', 'quantity'],
      dtype='object')
```

2.2.1 Price Standarized

In [29]:

```
from sklearn.preprocessing import StandardScaler
price_scalar = StandardScaler()

price_scalar.fit(X_train['price'].values.reshape(-1,1))

X_train_price_norm = price_scalar.transform(X_train['price'].values.reshape(-1,1))
X_cv_price_norm =price_scalar.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_norm = price_scalar.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_cv_price_norm.shape, y_cv.shape)
print(X_test_price_norm.shape, y_test.shape)
```

```
print("="*100)
```

```
After vectorizations
(28125, 1) (28125,)
(9375, 1) (9375,)
(12500, 1) (12500,)
====================================================================================================
```

2.2.2 Teacher_number_of_previously_posted_projects standardized

In [30]:

```
from sklearn.preprocessing import StandardScaler
price_scalar = StandardScaler()

price_scalar.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

X_train_pp_norm = price_scalar.transform(X_train['teacher_number_of_previously_posted_projects'].v
alues.reshape(-1,1))
X_cv_pp_norm =price_scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].values.r
eshape(-1,1))
X_test_pp_norm =
price_scalar.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))


print("After vectorizations")
print(X_train_pp_norm.shape, y_train.shape)
print(X_cv_pp_norm.shape, y_cv.shape)
print(X_test_pp_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(28125, 1) (28125,)
(9375, 1) (9375,)
(12500, 1) (12500,)
====================================================================================================
```

2.2.3 Quantity Standarized

In [31]:

```
from sklearn.preprocessing import StandardScaler
price_scalar = StandardScaler()

price_scalar.fit(X_train['quantity'].values.reshape(-1,1))

X_train_quantity_norm = price_scalar.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_quantity_norm =price_scalar.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_quantity_norm = price_scalar.transform(X_test['quantity'].values.reshape(-1,1))

print(X_train_quantity_norm.shape, y_train.shape)
print(X_cv_quantity_norm.shape, y_cv.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

```
(28125, 1) (28125,)
(9375, 1) (9375,)
(12500, 1) (12500,)
====================================================================================================
```

2.3 Vectorizing Categorical features

2.3.1 Vectorizing School_state

In [32]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(28125, 51) (28125,)
(9375, 51) (9375,)
(12500, 51) (12500,)
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'k
s', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd', 'ne', 'nh', 'nj', 'nm',
'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv
', 'wy']
====================================================================================================
```

2.3.2 Vectorizing teacher_prefix

In [33]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer.transform(X_train['teacher_prefix'].values)
X_cv_teacher_ohe = vectorizer.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(28125, 5) (28125,)
(9375, 5) (9375,)
(12500, 5) (12500,)
['dr', 'mr', 'mrs', 'ms', 'teacher']
====================================================================================================
```

2.3.3 Vectorizing grade_category

In [34]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['grade_category'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer.transform(X_train['grade_category'].values)
X_cv_grade_ohe = vectorizer.transform(X_cv['grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['grade_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
```

```
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(28125, 4) (28125,)
(9375, 4) (9375,)
(12500, 4) (12500,)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
====================================================================================================
```

2.3.4 Vectorizing clean_categories

In [35]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_cat_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_cv_cat_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_cat_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_cat_ohe.shape, y_train.shape)
print(X_cv_cat_ohe.shape, y_cv.shape)
print(X_test_cat_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(28125, 9) (28125,)
(9375, 9) (9375,)
(12500, 9) (12500,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language',
'math_science', 'music_arts', 'specialneeds', 'warmth']
====================================================================================================
```

2.3.5 Vectorizing clean_subcategories

In [36]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_scat_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_cv_scat_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_scat_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_scat_ohe.shape, y_train.shape)
print(X_cv_scat_ohe.shape, y_cv.shape)
print(X_test_scat_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(28125, 30) (28125,)
(9375, 30) (9375,)
(12500, 30) (12500,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government',
'college_careerprep', 'communityservice', 'earlydevelopment', 'economics', 'environmentalscience',
'esl', 'extracurricular', 'financialliteracy', 'foreignlanguages', 'gym_fitness',
'health_lifescience', 'health_wellness', 'history_geography', 'literacy', 'literature_writing', 'm
athematics', 'music', 'nutritioneducation', 'other', 'parentinvolvement', 'performingarts', 'socia
lsciences', 'specialneeds', 'teamsports', 'visualarts', 'warmth']
====================================================================================================
```

2.4 Encoding categorical & numerical features

In [37]:

```
#combining all the numerical and categorical values togeather

from scipy.sparse import hstack
X_tr_com =
hstack((X_train_cat_ohe,X_train_scat_ohe,X_train_state_ohe,X_train_teacher_ohe,X_train_grade_ohe,X
_train_pp_norm,X_train_price_norm,X_train_quantity_norm)).tocsr()
X_cr_com = hstack((X_cv_cat_ohe,X_cv_scat_ohe,X_cv_state_ohe,X_cv_teacher_ohe,X_cv_grade_ohe,X_cv_p
p_norm,X_cv_price_norm,X_cv_quantity_norm)).tocsr()
X_te_com =
hstack((X_test_cat_ohe,X_test_scat_ohe,X_test_state_ohe,X_test_teacher_ohe,X_test_grade_ohe,X_test_
pp_norm,X_test_price_norm,X_test_quantity_norm)).tocsr()

print("Final Data matrix")
print(X_tr_com.shape, y_train.shape)
print(X_cr_com.shape, y_cv.shape)
print(X_te_com.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix
(28125, 102) (28125,)
(9375, 102) (9375,)
(12500, 102) (12500,)
====================================================================================================
```

In [38]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=5,ngram_range=(1,1), max_features=5000)
vectorizer.fit(X_train['combined'].values)
X_tfidf = vectorizer.transform(X_train['combined'].values)
print("After vectorizations")
print(X_tfidf.shape)
```

```
After vectorizations
(28125, 5000)
```

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

In [39]:

```
indices = np.argsort(vectorizer.idf_)[::-1]
features = vectorizer.get_feature_names()
top_n = 2000
top_features = [features[i] for i in indices[:top_n]]
print (top_features[0:50])
```

```
['lacrosse', 'drone', 'violin', 'squad', 'chickens', 'chicks', 'cancer', 'pottery', 'ceramics', 'x
ylophone', 'bots', 'scanner', 'vr', 'dolls', 'horseshoe', 'cricut', 'hockey', 'monitors',
'flight', 'dna', 'storyworks', 'tickets', 'calculus', 'bats', 'holocaust', 'oils', 'tags',
'reeds', 'guitars', 'veggies', 'cubbies', 'xylophones', 'clarinet', 'goggles', 'benches', 'orff',
'flipped', 'littlebits', 'aprons', 'miami', 'diary', 'chess', 'bones', 'cream', 'partitions', 'fla
g', 'golf', 'elephant', 'cafe', 'pedals']
```

In [40]:

```
type(top_features)
```

Out[40]:

```
list
```

2.6 Computing Co-occurance matrix

In [41]:

```
#https://github.com/blob/master/ASSIGNMENT%20-%20Task%201%20%20.ipynb
def cooccurrenceMatrix(sample_data, neighbour_num , list_words):

        # Storing all words with their indices in the dictionary
        corpus = dict()
        # List of all words in the corpus
        doc = []
        index = 0
        for sent in sample_data:
            for word in sent.split():
                doc.append(word)
                corpus.setdefault(word,[])
                corpus[word].append(index)
                index += 1

        # Co-occurrence matrix
        matrix = []
        # rows in co-occurrence matrix
        for row in list_words:
            # row in co-occurrence matrix
            temp = []
            # column in co-occurrence matrix
            for col in list_words :
                if( col != row):
                    # No. of times col word is in neighbourhood of row word
                    count = 0
                    # Value of neighbourhood
                    num = neighbour_num
                    # Indices of row word in the corpus
                    positions = corpus[row]
                    for i in positions:
                        if i<(num-1):
                            # Checking for col word in neighbourhood of row
                            if col in doc[i:i+num]:
                                count +=1
                        elif (i>=(num-1)) and (i<=(len(doc)-num)):
                            # Check col word in neighbour of row
                            if (col in doc[i-(num-1):i+1]) and (col in doc[i:i+num]):
                                count +=2
                            # Check col word in neighbour of row
                            elif (col in doc[i-(num-1):i+1]) or (col in doc[i:i+num]):
                                count +=1
                        else :
                            if (col in doc[i-(num-1):i+1]):
                                count +=1


                    # appending the col count to row of co-occurrence matrix
                    temp.append(count)
                else:
                    # Append 0 in the column if row and col words are equal
                    temp.append(0)
            # appending the row in co-occurrence matrix
            matrix.append(temp)
        # Return co-occurrence matrix
        return np.array(matrix)
```

In [42]:

```
co_occ_matrix = cooccurrenceMatrix(X_train['combined'],5,top_features)
```

In [43]:

```
print(co_occ_matrix.shape)
```

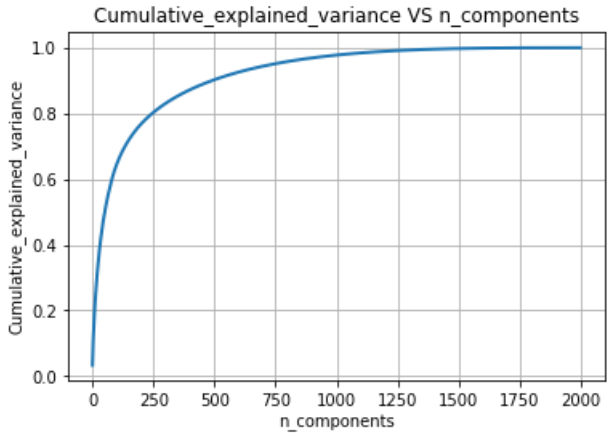
```
(2000, 2000)
```

In [44]:

```
def plotCumulativeVariance(co_occurrence_matrix):
        #Applying TruncatedSVD
        from sklearn.decomposition import TruncatedSVD
        max_features = co_occurrence_matrix.shape[1]-1
        svd = TruncatedSVD(n_components=max_features)
        svd_data = svd.fit_transform(co_occurrence_matrix)
        percentage_var_explained = svd.explained_variance_ / np.sum(svd.explained_variance_)
        cum_var_explained = np.cumsum(percentage_var_explained)
        # Plot the TrunvatedSVD 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.title("Cumulative_explained_variance VS n_components")
        plt.show()
```

In [45]:

```
plotCumulativeVariance(co_occ_matrix)
```



It can be observed that n_components as 250.

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

In [46]:

```
def computeVectors(co_occurrence_matrix, num_components):
        from sklearn.decomposition import TruncatedSVD
        svd_trunc = TruncatedSVD(n_components=num_components)
        svd_transform = svd_trunc.fit_transform(co_occurrence_matrix)
        # Returns Transformed matrix of Word-Vectors
        return svd_transform
```

In [47]:

```
word_vec_matrix =computeVectors(co_occ_matrix, 250)
print("Shape of word-vector : ",word_vec_matrix.shape)
```

```
Shape of word-vector :  (2000, 250)
```

2.8 Vectorizing the Text data

In [48]:

```
vectors_train = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['combined']): # for each review/sentence
    vector = np.zeros(250) # 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_features:
            i=top_features.index(word)
            vector += word_vec_matrix[i]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    vectors_train.append(vector)

print(len(vectors_train))

vectors_cv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['combined']): # for each review/sentence
    vector = np.zeros(250) # 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_features:
            i=top_features.index(word)
            vector += word_vec_matrix[i]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    vectors_cv.append(vector)

print(len(vectors_cv))

vectors_test = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['combined']): # for each review/sentence
    vector = np.zeros(250) # 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_features:
            i=top_features.index(word)
            vector += word_vec_matrix[i]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    vectors_test.append(vector)

print(len(vectors_test))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 28125/28125 [02:
34<00:00, 182.46it/s]
```

```
28125
```

```
100%|██████████████████████████████████████████████████████████████████████████| 9375/9375
[00:53<00:00, 174.13it/s]
```

```
9375
```

```
100%|██████████████████████████████████████████████████████████████████████████| 12500/12500 [01:
17<00:00, 170.21it/s]
```

```
12500
```

2.9 Merge the features from step 3 and step 4

In [49]:

```
X_final_train = hstack((X_tr_com,vectors_train)).tocsr()
X_final_cv = hstack((X_cr_com,vectors_cv)).tocsr()
```

```
X_final_test = hstack((X_te_com,vectors_test)).tocsr()
```

In [55]:

```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive 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 - 49041%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
```

2.10 Apply XGBoost on the Final Features from the above section

In [50]:

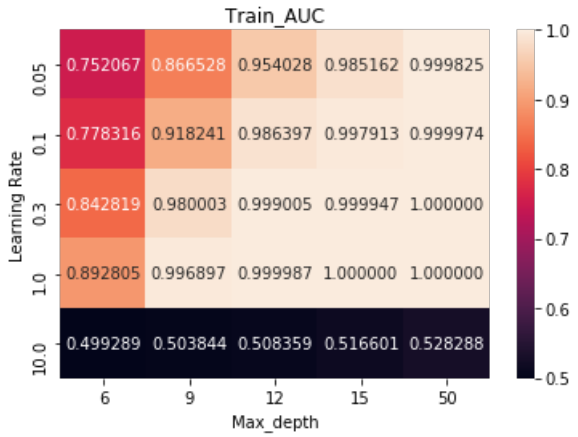
```
import sys
import math

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

import xgboost as xgb

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

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

In [70]:

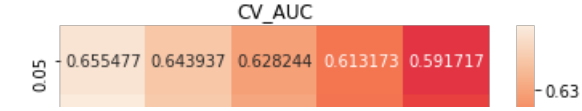
```
train_auc=[]
cv_auc=[]
clf = XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4)
#########################################################################
#                Change from here                                       #
#########################################################################
parameters = {
    #'num_boost_round': [100, 250, 500],
    'eta': [0.05, 0.1, 0.3, 1, 10],
    'max_depth': [6, 9, 12, 15, 50],
    #'subsample': [0.9, 1.0],
    #'colsample_bytree': [0.9, 1.0],
}

clf = GridSearchCV(clf, parameters,cv=3)
clf.fit(X_final_train, 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']
```

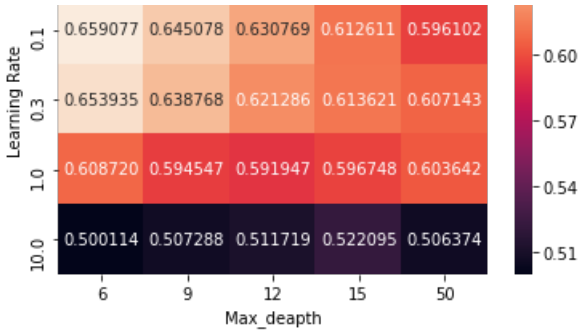
In [72]:

```
l=train_auc
ll=cv_auc
xx=[l[i:i+5] for i in range(0, len(l), 5)]
xxx=[ll[i:i+5] for i in range(0, len(ll), 5)]


import seaborn as sns

df_cm = pd.DataFrame(xx, index =parameters['eta'], columns =parameters['max_depth'])
sns.heatmap(df_cm, annot = True, fmt = "f")
plt.title("Train_AUC")
plt.xlabel("Max_depth")
plt.ylabel("Learning Rate")
plt.show()

print("="*100)
df_cm = pd.DataFrame(xxx,index =parameters['eta'], columns =parameters['max_depth'])
sns.heatmap(df_cm, annot = True, fmt = "f")
plt.title("CV_AUC")
plt.xlabel("Max_deapth")
plt.ylabel("Learning Rate")
plt.show()
```



==========



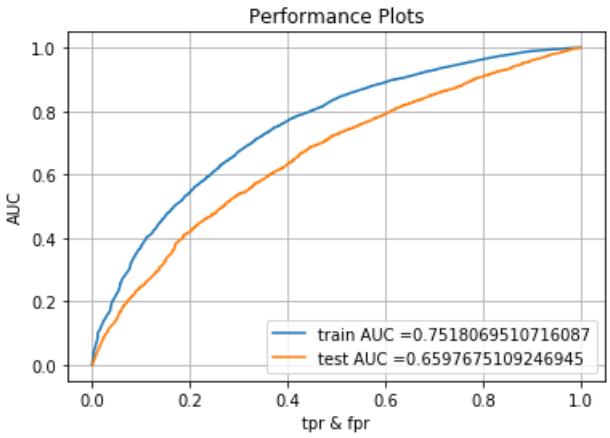


Obseravations

It can be observed that the Learing rate is 0.1 & max_depth is taken as 6.

In [74]:

```
from sklearn.metrics import roc_curve, auc


clf= XGBoostClassifier(eval_metric = 'auc', num_class = 2, nthread = 4, eta=0.1, max_depth=6)

clf.fit(X_final_train, y_train)


y_train_pred =batch_predict(clf,X_final_train)
y_test_pred = batch_predict(clf,X_final_test)

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_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("tpr & fpr")
plt.ylabel("AUC")
plt.title("Performance Plots")
plt.grid()
plt.show()
```



Observation

Train AUC is 0.751.
Test AUC is 0.659.
Learning rate is taken as 0.1
Max depth is taken as 6.

3. Conclusions

3.1 Obseravtions

Train AUC is 0.751.
Test AUC is 0.659.
Learning rate is taken as 0.1
Max depth is taken as 6.

3.2 Steps Taken

Step 1- We first Read the data from the file.
Step 2- Preprocessing all the data so that we can consider only information which has a value.
Step 3- Standardaried all the numerical data i.e Price,Quantity & Teacher number of previously posted projects.
Step 4- One Hot encoding of all the categorical data.
Step 5- Combining the data i.e Categorical and numerical.
Step 6- 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.
Step 7- Compute the co-occurance matrix with these 2k words, with window size=5.
Step 8- TruncatedSVD on calculated co-occurance matrix and reduce its dimensions, choose the number of components.
Step 9- Vectorize the essay text and project titles using these word vectors.
Step 10- Concatenate these truncatedSVD matrix, with the matrix with features.
Step 11- Apply GBDT on matrix that was formed and Hyperparameter tuning.
Step 12- Calculating the AUC value and plotting the heatmap.
Step 13- Drawing Conclusions0