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`

Feature	Description
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [1]:

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

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

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

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

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

from tqdm import tqdm
import os

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

```
C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\lib\site-packages\smart_open\ssh.py:34: Us
erWarning: paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip install paramiko`
to suppress
  warnings.warn('paramiko missing, opening SSH/SCP/SFTP paths will be disabled.  `pip install
paramiko` to suppress')
C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\lib\site-packages\gensim\utils.py:1197: Us
erWarning: 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('train_data.csv')
resource_data = pd.read_csv('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 (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

```
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[4]:

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

In [5]:

```
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 [6]:

```
#project_data=project_data.sample(n=4000)
```

In [7]:

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

```
Number of data points in train data (109248, 19)
--------------------------------------------------
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'
 'price' 'quantity']
```

1.2 preprocessing of `project_subject_categories`

In [8]:

```
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 [9]:

```
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]))
```

1.3 Text preprocessing

In [10]:

```
# 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 [11]:

In [11]:

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

Out[11]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [12]:

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

In [13]:

```
# 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)
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within our school.  Each student brings a wealth of knowledge and experiences to us that
open our eyes to new cultures, beliefs, and respect.\"The limits of your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
e is able to assist.  All families with students within the Level 1 proficiency status, will be a
offered to be a part of this program.  These educational videos will be specially chosen by the En
glish Learner 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 have access to a dvd player will have the
opportunity to check out a dvd player to use for the year.  The plan is to use these videos and ed
ucational dvd'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 l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f 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 bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents 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 hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay focused
on school.\r\n\r\nWhenever asked what the classroom is missing, my students always say more Hokki
Stools. They can't get their fill of the 5 stools we already have. When the students are sitting i
```

```
n group with me on the Hokki Stools, they are always moving, but at the same time doing their
work. Anytime the students get to pick where they can sit, the Hokki Stools are the first to be ta
ken. 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 th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing them to activate their core muscles for balance while they sit. For many of my students,
these chairs will take away the barrier 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 with plain walls, rows of
desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work
hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in
Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq
ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage
r learners; they are like sponges, absorbing all the information and experiences and keep on wanti
ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin
g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to
be one of a themed nautical environment. Creating a classroom environment is very important in the
success in each and every child's education. The nautical photo props will be used with each child
as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic
tures of each child with them, have them developed, and then hung in our classroom ready for their
first day of 4th grade.  This kind gesture will set the tone before even the first day of school!
The nautical thank you cards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of
my own pocket on resources to get our classroom ready. Please consider helping with this project t
o make our new school year a very successful one. Thank you!nannan
==================================================
```

In [14]:

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

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

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

In [15]:

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

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative,
and hard-working.  They are all unique - unique in their interests, their learning, their
abilities, and so much more.  What they all have in common is their desire to learn each day,
despite difficulties that they encounter.  \r\nOur classroom is amazing - because we understand
that everyone learns at their own pace.  As the teacher, I pride myself in making sure my students
are always engaged, motivated, and inspired to create their own learning! \r\nThis project is to h
elp my students choose seating that is more appropriate for them, developmentally.  Many students
tire of sitting in chairs during lessons, and having different seats available helps to keep them
engaged and learning.\r\nFlexible seating is important in our classroom, as many of our students s
truggle with attention, focus, and engagement.  We currently have stability balls for seating, as
well as regular chairs, but these stools will help students who have trouble with balance, or find
it difficult to sit on a stability ball for a long period of time.  We are excited to try these st
ools as a part of our engaging classroom community!nannan
==================================================
```

In [16]:

```
# \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)
```

```
Describing my students is not an easy task.  Many would say that they are inspirational, creative,
and hard-working.  They are all unique - unique in their interests, their learning, their
abilities, and so much more.  What they all have in common is their desire to learn each day,
despite difficulties that they encounter.    Our classroom is amazing - because we understand that
everyone learns at their own pace.  As the teacher, I pride myself in making sure my students are
always engaged, motivated, and inspired to create their own learning!   This project is to help my
students choose seating that is more appropriate for them, developmentally.  Many students tire of
sitting in chairs during lessons, and having different seats available helps to keep them engaged
and learning.  Flexible seating is important in our classroom, as many of our students struggle wi
th attention, focus, and engagement.  We currently have stability balls for seating, as well as re
gular chairs, but these stools will help students who have trouble with balance, or find it diffic
ult to sit on a stability ball for a long period of time.  We are excited to try these stools as a
part of our engaging classroom community!nannan
```

In [17]:

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

```
Describing my students is not an easy task Many would say that they are inspirational creative and
hard working They are all unique unique in their interests their learning their abilities and so m
uch more What they all have in common is their desire to learn each day despite difficulties that
they encounter Our classroom is amazing because we understand that everyone learns at their own pa
ce As the teacher I pride myself in making sure my students are always engaged motivated and inspi
red to create their own learning This project is to help my students choose seating that is more a
ppropriate for them developmentally Many students tire of sitting in chairs during lessons and hav
ing different seats available helps to keep them engaged and learning Flexible seating is
important in our classroom as many of our students struggle with attention focus and engagement We
currently have stability balls for seating as well as regular chairs but these stools will help st
udents who have trouble with balance or find it difficult to sit on a stability ball for a long pe
riod of time We are excited to try these stools as a part of our engaging classroom community nann
an
```

In [18]:

```
# 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', "do
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 [19]:

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

```
100%|████████████████████████████████████████████████████████████████████████| 109248/109248
[02:27<00:00, 739.51it/s]
```

In [20]:

```
# after preprocesing
preprocessed_essays[2000]
```

Out[20]:

```
'describing students not easy task many would say inspirational creative hard working they unique
unique interests learning abilities much what common desire learn day despite difficulties
encounter our classroom amazing understand everyone learns pace as teacher i pride making sure stu
dents always engaged motivated inspired create learning this project help students choose seating
appropriate developmentally many students tire sitting chairs lessons different seats available he
lps keep engaged learning flexible seating important classroom many students struggle attention fo
cus engagement we currently stability balls seating well regular chairs stools help students
trouble balance find difficult sit stability ball long period time we excited try stools part enga
ging classroom community nannan'
```

In [21]:

```
project_data['clean_essays'] = preprocessed_essays
project_data.drop(['essay'], axis=1, inplace=True)
```

1.4 Preprocessing of `project_title`

In [22]:

```
# similarly you can preprocess the titles also
```

In [23]:

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

```
100%|██████████████████████████████████████████████████████████████████████| 109248/109248
[00:06<00:00, 16500.70it/s]
```

In [24]:

```
project_data['clean_project_titles'] = preprocessed_titles
project_data.drop(['project_title'], axis=1, inplace=True)
project_data.head(2)
```

Out[24]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	pro
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Gra
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Gra

In [25]:

```
#https://planspace.org/20150607-textblob_sentiment/
#https://stackoverflow.com/questions/43485469/apply-textblob-in-for-each-row-of-a-dataframe
def cal_sentiment_polarity(inputString):
    try:
        return TextBlob(inputString).sentiment.polarity
    except:
        return 0
```

1.4.1 Calculating Sentiment score for essays`

In [26]:

```
project_data["essay_sentiment"]=project_data["clean_essays"].apply(cal_sentiment_polarity)
```

1.4.2 Calculating word counts

In [27]:

```
import re
def count_words(inputString):
    try:
        return len(re.findall(r'\w+', inputString))
    except:
        return None
```

In [28]:

```
project_data["title_word_count"]=project_data["clean_project_titles"].apply(count_words)
```

In [29]:

```
project_data["combine_essay_word_count"]=project_data["clean_essays"].apply(count_words)
```

1.5 Preparing data for models

In [30]:

```
project_data.columns
```

Out[30]:

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'price', 'quantity', 'clean_categories', 'clean_subcategories',
       'clean_essays', 'clean_project_titles', 'essay_sentiment',
       'title_word_count', 'combine_essay_word_count'],
      dtype='object')
```

we are going to consider

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

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

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

2. TruncatedSVD

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**

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

In [31]:

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

In [32]:

```
project_data['essay_and_title'] = project_data['clean_essays'] +' '+
project_data['clean_project_titles']
```

In [33]:

```
tfidf_model = TfidfVectorizer()
tfidf_model.fit_transform(project_data['essay_and_title'].values)
indices = np.argsort(tfidf_model.idf_)[::-1]
features = tfidf_model.get_feature_names()
top_n = 2000
top_words = [features[i] for i in indices[:top_n]]
```

2.2 Computing Co-occurance matrix

In [34]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# 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 reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

Co occurence matrix for toy data.

In [99]:

```
sentences=["abc def ijk pqr",
"pqr klm opq",
"lmn pqr xyz abc def pqr abc"]
top_words= ["abc", "pqr", "def"]
print(" Co_Occurance Matrix ")
noofneigh = 2
array = np.array([[0 for x in range(len(top_words))] for x in range(len(top_words))])
df = pd.DataFrame(array, index=top_words, columns=top_words)
for sent in tqdm(sentences):
    z = sent.split(' ')
    for word in top_words:
        for x in [x for (x, y) in enumerate(z) if word in y]:
            neighs_list=(z[max(x-noofneigh,0):x+noofneigh+1])
            for nei in neighs_list:
                if nei != word:
                    try:
                        df.loc[word, nei] += 1
                    except:
                        pass
print(df)
```

```
 Co_Occurance Matrix
```

```
100%|████████████████████████████████████████████████████████████| 3/3 [00:
00<00:00, 504.08it/s]
```

```
     abc  pqr  def
abc    0    3    3
pqr    3    0    2
def    3    2    0
```

In [35]:

```
def Co_Occurance_Matrix(X_Text, Imp_Feat):
    print(" Co_Occurance Matrix ")
    noofneigh = 2
    array = np.array([[0 for x in range(Imp_Feat)] for x in range(Imp_Feat)])
    df = pd.DataFrame(array, index=top_words, columns=top_words)

    for sent in tqdm(X_Text['essay_and_title'].values):
        z = sent.split(' ')
        for word in top_words:
            for x in [x for (x, y) in enumerate(z) if word in y]:
                neighs_list=(z[max(x-noofneigh,0):x+noofneigh+1])
                for nei in neighs_list:
                    if nei != word:
                        try:
                            df.loc[word, nei] += 1
                        except:
                            pass
    return df
```

In [36]:

```
import feather
if os.path.isfile('CoOccurance.feather'):
    df=pd.read_feather('CoOccurance.feather')
else:
    df=Co_Occurance_Matrix(project_data,top_n)
    feather.write_dataframe(df, 'CoOccurance.feather')
```

```
 Co_Occurance Matrix
```

```
100%|████████████████████████████████████████████████████| 109248/109248
[2:17:00<00:00, 13.29it/s]
```

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

In [83]:

```
# please write all the code with proper documentation, and proper titles for each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# 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 reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

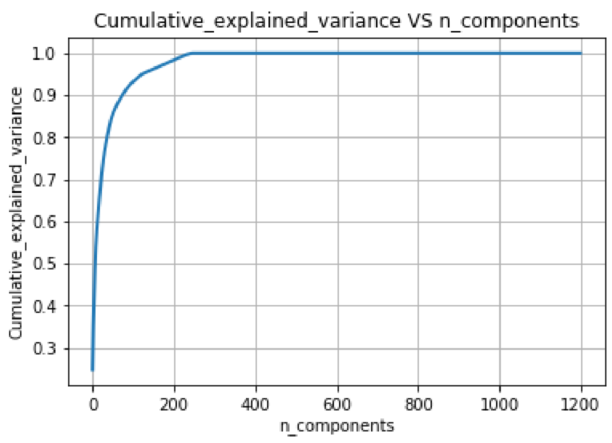
In [39]:

```
from sklearn.decomposition import TruncatedSVD
def SVD_Truncated(coo_matrix, no_of_comp):
    global Max_svd
    MaxVar = -1 # Max Explained varience

    Max_svd = 0 # initially 0

    #To get SVD with Max Explained varience

    for n_comp in no_of_comp:

        svd_matrix = TruncatedSVD(n_components=n_comp)
        svd=svd_matrix.fit(coo_matrix)
        exp_sum = svd.explained_variance_ratio_.sum()

        if exp_sum >  MaxVar :
            Max_svd = svd
            MaxVar  = exp_sum

    print("MaxExp==" ,MaxVar )
    percentage_var_explained =  Max_svd .explained_variance_ / np.sum( Max_svd .explained_variance_
)
    cum_var_explained = np.cumsum(percentage_var_explained)

    # Plotting for  MaxExp value in  list_component
    plt.plot(cum_var_explained , linewidth=2)
    plt.grid()
    plt.xlabel('n_components')
    plt.ylabel('Cumulative_explained_variance')
    plt.title("Cumulative_explained_variance VS n_components")
    plt.show()

    Max_svd=svd.transform(coo_matrix)
```

In [40]:

```
SVD_Truncated(df,[100,500,800,1000,1200,1500])
```

```
MaxExp== 1.0000000000000036
```

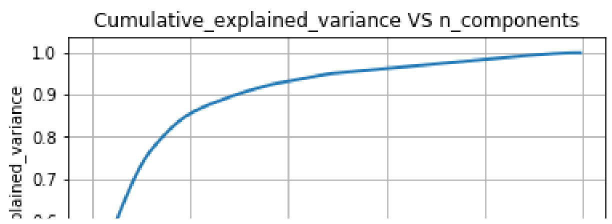


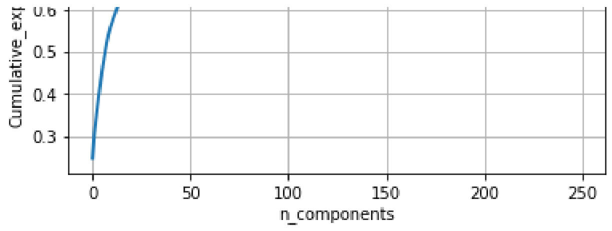
For n_components =250 the variance explained is maximum of 1.000%.So we can use n_components as 250 in further stages.

In [41]:

```
SVD_Truncated(df,[250])
```

```
MaxExp== 1.0000000000000038
```





In [87]:

```
#https://stackoverflow.com/questions/50915223/how-to-get-the-vector-representation-of-a-word-using
-a-trained-svd-model
#https://www.analyticsvidhya.com/blog/2018/10/stepwise-guide-topic-modeling-latent-semantic-analys
is/
#https://www.kaggle.com/dex314/tfidf-truncatedsvd-and-light-gbm
```

In [42]:

```
from sklearn import model_selection
X=project_data.drop(['project_is_approved'],axis=1)
y=project_data[['project_is_approved']]
X_tr, X_test, y_tr, y_test = model_selection.train_test_split(X, y, test_size=0.2, random_state=0,s
tratify=y)
```

In [45]:

```
print(X_tr.shape, y_tr.shape)
print(X_test.shape, y_test.shape)
```

```
(87398, 23) (87398, 1)
(21850, 23) (21850, 1)
```

In [90]:

```
tfidf = TfidfVectorizer(min_df=5, max_features=top_n)
tfidf.fit(X_tr['clean_essays'].values)
svdT = TruncatedSVD(n_components=100)
text_tfidf_tr = svdT.fit_transform(tfidf.transform(X_tr['clean_essays'].values))
text_tfidf_test= svdT.fit_transform(tfidf.transform(X_test['clean_essays'].values))
print(text_tfidf_tr.shape,text_tfidf_test.shape)
```

```
(32000, 100) (8000, 100)
```

In [80]:

```
col=list(df.columns)
def coo_word_vectors(phrase_values):
    avg_w2v_vectors = [];
    for sentence in tqdm(phrase_values): # 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 df.columns:
                vector += Max_svd[col.index(word)]
                cnt_words += 1
        if cnt_words != 0:
            vector /= cnt_words
        avg_w2v_vectors.append(vector)
    return avg_w2v_vectors
```

In [81]:

```
text_coo_vector_tr = coo_word_vectors(X_tr['clean_essays'].values)
text_coo_vector_test = coo_word_vectors(X_test['clean_essays'].values)
```

```
100%|██████████████████████████████████████████████████████████| 87398/87398
[00:07<00:00, 11181.02it/s]
100%|██████████████████████████████████████████████████████████| 21850/21850
```

```
[00:01<00:00, 11735.89it/s]
```

In [91]:

```
title_coo_vector_tr = coo_word_vectors(X_tr['clean_project_titles'].values)
title_coo_vector_test = coo_word_vectors(X_test['clean_project_titles'].values)
```

```
100%|██████████████████████████████████████████████████████████████████████| 87398/87398
[00:00<00:00, 210128.61it/s]
100%|██████████████████████████████████████████████████████████████████████| 21850/21850
[00:00<00:00, 204703.48it/s]
```

2.4 Merge the features from step 3 and step 4

In [92]:

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

1.5.1 Vectorizing Categorical data

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

In [83]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10)
categories_one_hot_tr = vectorizer.fit_transform(X_tr['clean_categories'].values)
categories_one_hot_test = vectorizer.transform(X_test['clean_categories'].values)
```

In [84]:

```
school_state_one_hot_tr = vectorizer.fit_transform(X_tr['school_state'].values)
school_state_one_hot_test = vectorizer.transform(X_test['school_state'].values)
```

In [95]:

```
# you can do the similar thing with state, teacher_prefix and project_grade_category also
```

In [85]:

```
sub_categories_one_hot_tr = vectorizer.fit_transform(X_tr['clean_subcategories'].values)
sub_categories_one_hot_test = vectorizer.transform(X_test['clean_subcategories'].values)
```

In [86]:

```
teacher_prefix_one_hot_tr = vectorizer.fit_transform(X_tr['teacher_prefix'].values.astype('U'))
teacher_prefix_one_hot_test = vectorizer.transform(X_test['teacher_prefix'].values.astype('U'))
```

In [87]:

```
project_grade_category_one_hot_tr = vectorizer.fit_transform(X_tr['project_grade_category'].values
)
project_grade_category_one_hot_test = vectorizer.transform(X_test['project_grade_category'].values
)
```

1.5.3 Vectorizing Numerical features

In [88]:

```
from sklearn.preprocessing import StandardScaler
price_scalar = StandardScaler()
price_standardized_tr = price_scalar.fit_transform(X_tr['price'].values.reshape(-1,1)) # finding th
e mean and standard deviation of this data
price_standardized_test = price_scalar.transform(X_test['price'].values.reshape(-1,1))
```

In [89]:

```
previously_posted_scalar = StandardScaler()
previously_posted_standardized_tr =
previously_posted_scalar.fit_transform(X_tr['teacher_number_of_previously_posted_projects'].values
.reshape(-1, 1))
previously_posted_standardized_test =
previously_posted_scalar.transform(X_test['teacher_number_of_previously_posted_projects'].values.r
eshape(-1, 1))
```

```
C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\utils\validation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\utils\validation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\utils\validation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

In [90]:

```
quantity_scalar = StandardScaler()
quantity_standardized_tr = quantity_scalar.fit_transform(X_tr['quantity'].values.reshape(-1, 1))
quantity_standardized_test = quantity_scalar.transform(X_test['quantity'].values.reshape(-1, 1))
```

```
C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\utils\validation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\utils\validation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.

C:\Users\mchetankumar\AppData\Local\Continuum\anaconda3\lib\site-
packages\sklearn\utils\validation.py:595: DataConversionWarning:

Data with input dtype int64 was converted to float64 by StandardScaler.
```

1.5.4 Merging all the above features

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

In [92]:

```
from scipy.sparse import hstack
set1_tr =
hstack((categories_one_hot_tr,sub_categories_one_hot_tr,teacher_prefix_one_hot_tr,school_state_one_
hot_tr,project_grade_category_one_hot_tr,price_standardized_tr,previously_posted_standardized_tr,q
uantity_standardized_tr,text_coo_vector_tr,title_coo_vector_tr))
```

```
set1_test =
hstack((categories_one_hot_test,sub_categories_one_hot_test,teacher_prefix_one_hot_test,school_stat
e_one_hot_test,project_grade_category_one_hot_test,price_standardized_test,previously_posted_standa
rdized_test,quantity_standardized_test,text_coo_vector_test,title_coo_vector_test))
print(set1_tr.shape)
print(set1_test.shape)
```

```
(87398, 600)
(21850, 600)
```

In [0]:

```
import sys
import math

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

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

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

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


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)
```

```
Y = np.array([0, 1, 0, 1, 0, 1])
clf.fit(X, Y)

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

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

2.5 Apply XGBoost on the Final Features from the above section

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

In [93]:

```
import xgboost as xgb
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import confusion_matrix
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
```

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 each subsection
# go through documentations and blogs before you start coding
# first figure out what to do, and then think about how to do.
# reading and understanding error messages will be very much helpfull in debugging your code
# when you plot any graph make sure you use
    # a. Title, that describes your plot, this will be very helpful to the reader
    # b. Legends if needed
    # c. X-axis label
    # d. Y-axis label
```

In [94]:

```
xgb_model  = xgb.XGBClassifier(class_weight='balanced',n_jobs=-1)
parameters = [{'n_estimators': [10,20,50,70],'max_depth':   [4,5,6,7]}]
grid_search = GridSearchCV(xgb_model, parameters, cv=5, scoring='roc_auc')
grid_search.fit(set1_tr, y_tr.values.ravel())

scores_train = grid_search.cv_results_['mean_train_score'].reshape(len(parameters[0]['n_estimators'
]),len(parameters[0]['max_depth']))
scores_cv = grid_search.cv_results_['mean_test_score'].reshape(len(parameters[0]['n_estimators']),l
en(parameters[0]['max_depth']))

trace1 = go.Scatter3d(x=grid_search.cv_results_['param_n_estimators'],y=grid_search.cv_results_['pa
ram_max_depth'],z=scores_train.ravel(), name = 'train')
trace2 = go.Scatter3d(x=grid_search.cv_results_['param_n_estimators'],y=grid_search.cv_results_['pa
ram_max_depth'],z=scores_cv.ravel(), name = 'Cross validation')
data = [trace1, trace2]

layout = go.Layout(scene = dict(
        xaxis = dict(title='n_estimators'),
        yaxis = dict(title='max_depth'),
        zaxis = dict(title='AUC'),))
```

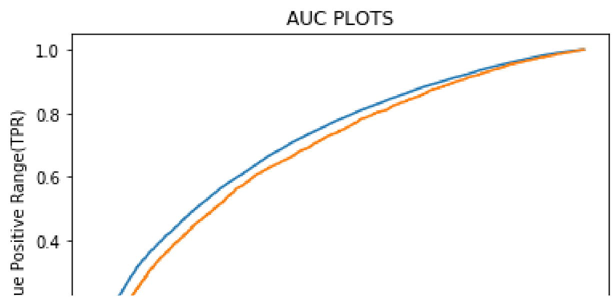
```
fig = go.Figure(data=data, layout=layout)
offline.iplot(fig, filename='3d-scatter-colorscale')
```

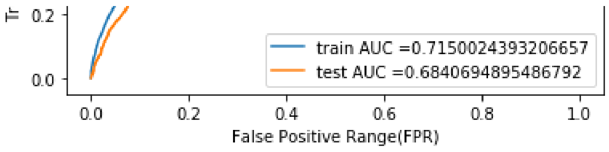
In [95]:

```
xg_best_params={'max_depth': 6, 'n_estimators':50}
```

In [96]:

```
xgb_model =
xgb.XGBClassifier(max_depth=xg_best_params['max_depth'],n_estimators=xg_best_params['n_estimators'
],class_weight='balanced',n_jobs=-1)
xgb_model.fit(set1_tr, np.ravel(y_tr,order='C'))
y_train_pred=[]
y_test_pred=[]
for j in range(0, set1_tr.shape[0], 1000):
    y_train_pred.extend(xgb_model.predict_proba(set1_tr.tocsr()[j:j+1000])[:,1])
for j in range(0, set1_test.shape[0], 1000):
    y_test_pred.extend(xgb_model.predict_proba(set1_test.tocsr()[j:j+1000])[:,1])
train_fpr, train_tpr, thresholds =  roc_curve(y_tr, y_train_pred)
test_fpr, test_tpr, 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)))
xg_train_auc=auc(train_fpr, train_tpr)
xg_test_auc=auc(test_fpr, test_tpr)
plt.legend()
plt.xlabel("False Positive Range(FPR)")
plt.ylabel("True Positive Range(TPR)")
plt.title("AUC PLOTS")
plt.show()
```





Tr
0.2
0.0
train AUC =0.7150024393206657
test AUC =0.6840694895486792
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Range(FPR)