DonorsChoose

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

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-6bn6qk8qdgf4n4g3pfee
6491hc0brc4i.apps.googleusercontent.com&redirect_uri=urn%3aietf%3awg%3aoauth%3a2.0%3aoob&response_type=code&scope=
email%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdocs.test%20https%3a%2f%2fwww.googleapis.com%2fauth%2fdrive%20ht
tps%3a%2f%2fwww.googleapis.com%2fauth%2fdrive.photos.readonly%20https%3a%2f%2fwww.googleapis.com%2fauth%2fpeopleap
i.readonly

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

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 (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **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*
`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 [2]:
```python
%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
!pip install chart_studio
from chart_studio import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

```
Collecting chart_studio
  Downloading https://files.pythonhosted.org/packages/ca/ce/330794a6b6ca4b9182c38fc69dd2a9cbff60fd49421cb8648ee5fe
e352dc/chart_studio-1.1.0-py3-none-any.whl (64kB)
     |████████████████████████████████| 71kB 4.9MB/s
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from chart_studio) (1.12.0)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from chart_studio) (1.3.
3)
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (from chart_studio) (4.4.1)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from chart_studio) (2.23.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.6/dist-packages
(from requests->chart_studio) (1.24.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-packages (from requests->chart_studi
o) (2.9)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (from requests->chart_s
tudio) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (from requests->chart_
studio) (2020.4.5.1)
Installing collected packages: chart-studio
Successfully installed chart-studio-1.1.0
```

1.1 Loading Data

In [0]:
```
project_data = pd.read_csv('/content/drive/My Drive/Assignments_DonorsChoose_2018/train_data.csv')
resource_data = pd.read_csv('/content/drive/My Drive/Assignments_DonorsChoose_2018/resources.csv')
```

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

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' '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 [0]:
```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[0]:

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

1.2 Preprocessing Categorical Data

1.2.1 preprocessing `project_subject_categories`

In [0]:
```
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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&",
"Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'Th
e')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Scienc
e"
        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]))
```

In [0]:
```
sorted_cat_dict.keys()
```

Out[0]:
```
dict_keys(['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Spo
rts', 'Math_Science', 'Literacy_Language'])
```

1.2.2 preprocessing of `project_subject_subcategories`

In [0]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://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 & Hunger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Science"=> "Math","&",
"Science"
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''(i.e removing 'Th
e')
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Science"=>"Math&Scienc
e"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    sub_cat_list.append(temp.strip())

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

from collections import Counter
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]))
```

In [0]:
```
sorted_sub_cat_dict.keys()
```

Out[0]:
```
dict_keys(['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_G
overnment', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts',
'CharacterEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScien
ce', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSc
iences', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy'])
```

1.2.3 preprocessing of `School State`

In [0]:
```
project_data['school_state'].unique()
```

Out[0]:
```
array(['IN', 'FL', 'AZ', 'KY', 'TX', 'CT', 'GA', 'SC', 'NC', 'CA', 'NY',
       'OK', 'MA', 'NV', 'OH', 'PA', 'AL', 'LA', 'VA', 'AR', 'WA', 'WV',
       'ID', 'TN', 'MS', 'CO', 'UT', 'IL', 'MI', 'HI', 'IA', 'RI', 'NJ',
       'MO', 'DE', 'MN', 'ME', 'WY', 'ND', 'OR', 'AK', 'MD', 'WI', 'SD',
       'NE', 'NM', 'DC', 'KS', 'MT', 'NH', 'VT'], dtype=object)
```

In [0]:
```
project_data['school_state'][project_data['school_state'].isnull()==True]
```

Out[0]:
```
Series([], Name: school_state, dtype: object)
```

In [0]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['school_state'].values:
    my_counter.update(word.split())

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

In [0]:
```
sorted_school_state_dict.keys()
```

Out[0]:
```
dict_keys(['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'IA',
'ID', 'AR', 'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA',
'MA', 'LA', 'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX', 'CA'])
```

1.2.4 preprocessing of `Teacher Prefix`

In [0]:
```
project_data.groupby(['teacher_prefix'])['teacher_prefix'].count()
```

Out[0]:
```
teacher_prefix
Dr.           13
Mr.        10648
Mrs.       57269
Ms.        38955
Teacher     2360
Name: teacher_prefix, dtype: int64
```

```
In [0]: project_data['teacher_prefix'][project_data['teacher_prefix'].isnull()==True]
```

```
Out[0]: 7820     NaN
        30368    NaN
        57654    NaN
        Name: teacher_prefix, dtype: object
```

```
In [0]: project_data['teacher_prefix'].fillna(project_data['teacher_prefix'].mode()[0],inplace=True)
```

```
In [0]: project_data['teacher_prefix'][project_data['teacher_prefix'].isnull()==True]
```

```
Out[0]: Series([], Name: teacher_prefix, dtype: object)
```

```
In [0]: project_data['teacher_prefix'].unique()
```

```
Out[0]: array(['Mrs.', 'Mr.', 'Ms.', 'Teacher', 'Dr.'], dtype=object)
```

```
In [0]: teacher_prefix = list(project_data['teacher_prefix'].values)

        teacher_prefix_list = []
        for i in teacher_prefix:
            temp = ""
            temp = i.split('.')
            temp = i.replace('.','')
            teacher_prefix_list.append(temp)

        project_data['clean_teacher_prefix'] = teacher_prefix_list
        project_data.drop(['teacher_prefix'], 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_teacher_prefix'].values:
            my_counter.update(word.split())

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

```
In [0]: sorted_teacher_prefix_dict.keys()
```

```
Out[0]: dict_keys(['Dr', 'Teacher', 'Mr', 'Ms', 'Mrs'])
```

```
In [0]: project_data.groupby(['clean_teacher_prefix'])['clean_teacher_prefix'].count()
```

```
Out[0]: clean_teacher_prefix
        Dr            13
        Mr         10648
        Mrs        57272
        Ms         38955
        Teacher     2360
        Name: clean_teacher_prefix, dtype: int64
```

1.2.5 preprocessing of `Project Grade Category`

```
In [0]: project_data.groupby(['project_grade_category'])['project_grade_category'].count()
```

```
Out[0]: project_grade_category
        Grades 3-5       37137
        Grades 6-8       16923
        Grades 9-12      10963
        Grades PreK-2    44225
        Name: project_grade_category, dtype: int64
```

```
In [0]: project_data['project_grade_category'][project_data['project_grade_category'].isnull()==True]
```

```
Out[0]: Series([], Name: project_grade_category, dtype: object)
```

In [0]:
```
project_grade_category = list(project_data['project_grade_category'].values)

project_grade_category_list = []
for i in project_grade_category:
    temp = ""
    temp = i.split(' ')
    temp = i.replace('Grades ','')
    project_grade_category_list.append(temp)

project_data['clean_project_grade_category'] = project_grade_category_list
project_data.drop(['project_grade_category'], 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_project_grade_category'].values:
    my_counter.update(word.split())

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

In [0]:
```
sorted_project_grade_category_dict.keys()
```

Out[0]:
```
dict_keys(['9-12', '6-8', '3-5', 'PreK-2'])
```

In [0]:
```
project_data.groupby(['clean_project_grade_category'])['clean_project_grade_category'].count()
```

Out[0]:
```
clean_project_grade_category
3-5       37137
6-8       16923
9-12      10963
PreK-2    44225
Name: clean_project_grade_category, dtype: int64
```

1.3 Text Preprocessing of project_essay

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

In [0]:
```
project_data.head(1)
```

Out[0]:

	Unnamed: 0	id	teacher_id	school_state	project_submitted_datetime	project_title	project_essay_1	project_essay
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	IN	2016-12-05 13:43:57	Educational Support for English Learners at Home	My students are English learners that are work...	\"The limits your langua are the limits

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

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

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

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

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

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

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

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

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

In [0]:
```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'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', 'afte
r',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'furt
her',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'm
ore',\
            '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', 'r
e', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',
\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "we
ren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [0]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [01:02<00:00, 1751.57it/s]
```

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

Out[0]:
```
'my kindergarten students varied disabilities ranging speech language delays cognitive delays gross fine motor del
ays autism they eager beavers always strive work hardest working past limitations the materials ones i seek studen
ts i teach title i school students receive free reduced price lunch despite disabilities limitations students love
coming school come eager learn explore have ever felt like ants pants needed groove move meeting this kids feel ti
me the want able move learn say wobble chairs answer i love develop core enhances gross motor turn fine motor skil
ls they also want learn games kids not want sit worksheets they want learn count jumping playing physical engageme
nt key success the number toss color shape mats make happen my students forget work fun 6 year old deserves nanna
n'
```

In [0]:
```
project_data['preprocessed_essays'] = preprocessed_essays
project_data.drop(['essay'], axis=1, inplace=True)
```

1.4 Preprocessing of `project_title`

In [0]:
```
project_data['project_title'][2000:2010]
```

Out[0]:
```
2000                    Steady Stools for Active Learning
2001                                   Classroom Supplies
2002    Kindergarten Students Deserve Quality  Books a...
2003                                Listen to Understand!
2004                             iPads to iGnite Learning
2005                                 Tablets For Learning
2006                                             Go P.E.!
2007                                 Making Learning Fun!
2008    Empowerment Through Silk Screen Designed Tee S...
2009                                 Let's Play Together!
Name: project_title, dtype: object
```

In [0]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:02<00:00, 41204.32it/s]
```

In [0]:
```
preprocessed_titles[2000:2010]
```

Out[0]:
```
['steady stools active learning',
 'classroom supplies',
 'kindergarten students deserve quality books vibrant rug',
 'listen understand',
 'ipads ignite learning',
 'tablets for learning',
 'go p e',
 'making learning fun',
 'empowerment through silk screen designed tee shirts',
 'let play together']
```

In [0]:
```
project_data['preprocessed_titles'] = preprocessed_titles
project_data.drop(['project_title'], axis=1, inplace=True)
```

1.5 Merging Numerical data in Resources to project_data

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

```
In [0]: project_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 109248 entries, 0 to 109247
Data columns (total 20 columns):
 #   Column                                        Non-Null Count   Dtype  
---  ------                                        --------------   -----  
 0   Unnamed: 0                                    109248 non-null  int64  
 1   id                                            109248 non-null  object 
 2   teacher_id                                    109248 non-null  object 
 3   school_state                                  109248 non-null  object 
 4   project_submitted_datetime                    109248 non-null  object 
 5   project_essay_1                               109248 non-null  object 
 6   project_essay_2                               109248 non-null  object 
 7   project_essay_3                               3758 non-null    object 
 8   project_essay_4                               3758 non-null    object 
 9   project_resource_summary                      109248 non-null  object 
 10  teacher_number_of_previously_posted_projects  109248 non-null  int64  
 11  project_is_approved                           109248 non-null  int64  
 12  clean_categories                              109248 non-null  object 
 13  clean_subcategories                           109248 non-null  object 
 14  clean_teacher_prefix                          109248 non-null  object 
 15  clean_project_grade_category                  109248 non-null  object 
 16  preprocessed_essays                           109248 non-null  object 
 17  preprocessed_titles                           109248 non-null  object 
 18  price                                         109248 non-null  float64
 19  quantity                                      109248 non-null  int64  
dtypes: float64(1), int64(4), object(15)
memory usage: 17.5+ MB
```

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
- Essay : text data

- quantity : numerical
- teacher_number_of_previously_posted_projects : numerical
- price : numerical
```

```
In [0]: data1 = project_data.drop(['Unnamed: 0', 'id','project_submitted_datetime','project_essay_1','project_essay_2','project_essay_3','project_essay_4','project_resource_summary','teacher_id'], axis = 1)
```

```
In [0]: data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 109248 entries, 0 to 109247
Data columns (total 11 columns):
 #   Column                                        Non-Null Count   Dtype  
---  ------                                        --------------   -----  
 0   school_state                                  109248 non-null  object 
 1   teacher_number_of_previously_posted_projects  109248 non-null  int64  
 2   project_is_approved                           109248 non-null  int64  
 3   clean_categories                              109248 non-null  object 
 4   clean_subcategories                           109248 non-null  object 
 5   clean_teacher_prefix                          109248 non-null  object 
 6   clean_project_grade_category                  109248 non-null  object 
 7   preprocessed_essays                           109248 non-null  object 
 8   preprocessed_titles                           109248 non-null  object 
 9   price                                         109248 non-null  float64
 10  quantity                                      109248 non-null  int64  
dtypes: float64(1), int64(3), object(7)
memory usage: 10.0+ MB
```

1.6 Adding new features

1. essays_word_count
2. title_word_count
3. Combine preprocessed Essays and project titles
4. Sentiment Scores

In [0]:
```
data1["essays_word_count"] = data1['preprocessed_essays'].str.count(" ")+1
data1["title_word_count"] = data1['preprocessed_titles'].str.count(" ")+1
```

In [0]:
```
# combining preprocessed essays and project_titles
data1["preprocessed_title_essays"] = data1["preprocessed_essays"] + data1["preprocessed_titles"]
```

In [0]:
```
# for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students with the biggest
enthusiasm \
# for learning my students learn in many different ways using all of our senses and multiple intelligences i use a
 wide range\
# of techniques to help all my students succeed students in my class come from a variety of different backgrounds w
hich makes\
# for wonderful sharing of experiences and cultures including native americans our school is a caring community of
 successful \
# learners which can be seen through collaborative student project based learning in and out of the classroom kinde
rgarteners \
# in my class love to work with hands on materials and have many different opportunities to practice a skill before
it is\
# mastered having the social skills to work cooperatively with friends is a crucial aspect of the kindergarten curr
iculum\
# montana is the perfect place to learn about agriculture and nutrition my students love to role play in our preten
d kitchen\
# in the early childhood classroom i have had several kids ask me can we try cooking with real food i will take the
ir idea \
# and create common core cooking lessons where we learn important math and writing concepts while cooking delicious
healthy \
# food for snack time my students will have a grounded appreciation for the work that went into making the food and
knowledge \
# of where the ingredients came from as well as how it is healthy for their bodies this project would expand our le
arning of \
# nutrition and agricultural cooking recipes by having us peel our own apples to make homemade applesauce make our
 own bread \
# and mix up healthy plants from our classroom garden in the spring we will also create our own cookbooks to be pri
nted and \
# shared with families students will gain math and literature skills as well as a life long enjoyment for healthy c
ooking \
# nannan'

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

In [0]:
```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
nltk.download('vader_lexicon')
sid = SentimentIntensityAnalyzer()

def calculate_sentiment_scores(string):
    ss = sid.polarity_scores(string)
    return ss["neg"], ss["neu"], ss["pos"], ss["compound"]

#https://stackoverflow.com/questions/16236684/apply-pandas-function-to-column-to-create-multiple-new-columns
data1["neg_score"], data1["neu_score"], data1["pos_score"], data1["compound_score"] = \
                                                              zip(*data1["preprocessed_essays"].map(calculate_senti
ment_scores))
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
```

Train test split

In [0]:
```
copy_x = data1.copy()
copy_y = data1['project_is_approved'].copy()

data1 = data1[:50000]
y = data1['project_is_approved'][:50000]
data1.shape, y.shape
```

Out[0]:
```
((50000, 18), (50000,))
```

In [0]:
```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data1, y, test_size=0.33, stratify=y)
```

In [0]:
```
#Features
X_train.drop(['project_is_approved'], axis=1, inplace=True)

X_test.drop(['project_is_approved'], axis=1, inplace=True)
```

In [0]:
```
X_train.head()
```

Out[0]:

	school_state	teacher_number_of_previously_posted_projects	clean_categories	clean_subcategories	clean_teacher_prefix	clean_project_
16875	NY	18	Math_Science Music_Arts	EnvironmentalScience VisualArts	Mrs	
18818	WV	66	Literacy_Language	Literacy	Mrs	
17086	CO	4	Health_Sports	Health_Wellness	Mrs	
40147	CA	0	AppliedLearning Literacy_Language	Extracurricular Literature_Writing	Ms	
8224	AL	0	Literacy_Language Math_Science	Literacy Mathematics	Teacher	

1.7 Make Data Model Ready: encoding numerical, categorical features

In [0]:
```
# 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
```

1.7.1 Numerical features

```
1. teacher_number_of_previously_posted_projects
2. price
3. quantity
4. essays word count
5. title word count
6. neg score
7. pos score
8. neu score
9. compound score
```

1.7.1.1 Teacher number of previously posted projects

In [0]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-1))

X_train_TPPP_norm = normalizer.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(1,-
1))
X_test_TPPP_norm = normalizer.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(1,-1
))

print("After vectorizations")
print(X_train_TPPP_norm.shape, y_train.shape)
print(X_test_TPPP_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 33500) (33500,)
(1, 16500) (16500,)
====================================================================================================
```

In [0]:
```
print("Transpose of teacher number of previously posted projects")

X_train_TPPP_norm = X_train_TPPP_norm.transpose()
X_test_TPPP_norm = X_test_TPPP_norm.transpose()

print("After transpose")
print(X_train_TPPP_norm.shape, y_train.shape)
print(X_test_TPPP_norm.shape, y_test.shape)
print("="*100)
```

```
Transpose of teacher number of previously posted projects
After transpose
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

1.7.1.2 price

In [0]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(1,-1))

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

print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 33500) (33500,)
(1, 16500) (16500,)
====================================================================================================
```

In [0]:
```
print("Transpose of price")

X_train_price_norm = X_train_price_norm.transpose()
X_test_price_norm = X_test_price_norm.transpose()


print("After vectorizations")
print(X_train_price_norm.shape, y_train.shape)
print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
Transpose of price
After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

1.7.1.3 quantity

In [0]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['quantity'].values.reshape(1,-1))

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

print("After vectorizations")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 33500) (33500,)
(1, 16500) (16500,)
====================================================================================================
```

In [0]:
```
print("Transpose of Quantity")

X_train_quantity_norm = X_train_quantity_norm.transpose()
X_test_quantity_norm = X_test_quantity_norm.transpose()


print("After vectorizations")
print(X_train_quantity_norm.shape, y_train.shape)
print(X_test_quantity_norm.shape, y_test.shape)
print("="*100)
```

```
Transpose of Quantity
After vectorizations
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

1.7.1.4 Essay Word count

In [0]:
```
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 33500 entries, 16875 to 37981
Data columns (total 17 columns):
 #   Column                                        Non-Null Count  Dtype  
---  ------                                        --------------  -----  
 0   school_state                                  33500 non-null  object 
 1   teacher_number_of_previously_posted_projects  33500 non-null  int64  
 2   clean_categories                              33500 non-null  object 
 3   clean_subcategories                           33500 non-null  object 
 4   clean_teacher_prefix                          33500 non-null  object 
 5   clean_project_grade_category                  33500 non-null  object 
 6   preprocessed_essays                           33500 non-null  object 
 7   preprocessed_titles                           33500 non-null  object 
 8   price                                         33500 non-null  float64
 9   quantity                                      33500 non-null  int64  
 10  essays_word_count                             33500 non-null  int64  
 11  title_word_count                              33500 non-null  int64  
 12  neg_score                                     33500 non-null  float64
 13  neu_score                                     33500 non-null  float64
 14  pos_score                                     33500 non-null  float64
 15  compound_score                                33500 non-null  float64
 16  preprocessed_title_essays                     33500 non-null  object 
dtypes: float64(5), int64(4), object(8)
memory usage: 4.6+ MB
```

In [0]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['essays_word_count'].values.reshape(1,-1))

X_train_esscnt_norm = normalizer.transform(X_train['essays_word_count'].values.reshape(1,-1))
X_test_esscnt_norm = normalizer.transform(X_test['essays_word_count'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_esscnt_norm.shape, y_train.shape)
print(X_test_esscnt_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 33500) (33500,)
(1, 16500) (16500,)
====================================================================================================
```

In [0]:
```
print("Transpose of Essay word counts")

X_train_esscnt_norm = X_train_esscnt_norm.transpose()
X_test_esscnt_norm = X_test_esscnt_norm.transpose()

print("After transpose")
print(X_train_esscnt_norm.shape, y_train.shape)
print(X_test_esscnt_norm.shape, y_test.shape)
print("="*100)
```

```
Transpose of Essay word counts
After transpose
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

1.7.1.5 Title Word count

In [0]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['title_word_count'].values.reshape(1,-1))

X_train_titlecnt_norm = normalizer.transform(X_train['title_word_count'].values.reshape(1,-1))
X_test_titlecnt_norm = normalizer.transform(X_test['title_word_count'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_titlecnt_norm.shape, y_train.shape)
print(X_test_titlecnt_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 33500) (33500,)
(1, 16500) (16500,)
====================================================================================================
```

In [0]:
```
print("Transpose of Title word counts")

X_train_titlecnt_norm = X_train_titlecnt_norm.transpose()
X_test_titlecnt_norm = X_test_titlecnt_norm.transpose()

print("After transpose")
print(X_train_titlecnt_norm.shape, y_train.shape)
print(X_test_titlecnt_norm.shape, y_test.shape)
print("="*100)
```

```
Transpose of Title word counts
After transpose
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

1.7.1.6 Neg Score

```
In [0]: from sklearn.preprocessing import Normalizer
        normalizer = Normalizer()
        # normalizer.fit(X_train['price'].values)
        # this will rise an error Expected 2D array, got 1D array instead:
        # array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
        # Reshape your data either using
        # array.reshape(-1, 1) if your data has a single feature
        # array.reshape(1, -1)  if it contains a single sample.
        normalizer.fit(X_train['neg_score'].values.reshape(1,-1))

        X_train_neg_norm = normalizer.transform(X_train['neg_score'].values.reshape(1,-1))
        X_test_neg_norm = normalizer.transform(X_test['neg_score'].values.reshape(1,-1))

        print("After vectorizations")
        print(X_train_neg_norm.shape, y_train.shape)
        print(X_test_neg_norm.shape, y_test.shape)
        print("="*100)
```

```
After vectorizations
(1, 33500) (33500,)
(1, 16500) (16500,)
====================================================================================================
```

```
In [0]: print("Transpose of Neg score")

        X_train_neg_norm = X_train_neg_norm.transpose()
        X_test_neg_norm = X_test_neg_norm.transpose()

        print("After transpose")
        print(X_train_neg_norm.shape, y_train.shape)
        print(X_test_neg_norm.shape, y_test.shape)
        print("="*100)
```

```
Transpose of Neg score
After transpose
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

1.7.1.7 Pos Score

```
In [0]: from sklearn.preprocessing import Normalizer
        normalizer = Normalizer()
        # normalizer.fit(X_train['price'].values)
        # this will rise an error Expected 2D array, got 1D array instead:
        # array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
        # Reshape your data either using
        # array.reshape(-1, 1) if your data has a single feature
        # array.reshape(1, -1)  if it contains a single sample.
        normalizer.fit(X_train['pos_score'].values.reshape(1,-1))

        X_train_pos_norm = normalizer.transform(X_train['pos_score'].values.reshape(1,-1))
        X_test_pos_norm = normalizer.transform(X_test['pos_score'].values.reshape(1,-1))

        print("After vectorizations")
        print(X_train_pos_norm.shape, y_train.shape)
        print(X_test_pos_norm.shape, y_test.shape)
        print("="*100)
```

```
After vectorizations
(1, 33500) (33500,)
(1, 16500) (16500,)
====================================================================================================
```

```
In [0]: print("Transpose of Pos score")

        X_train_pos_norm = X_train_pos_norm.transpose()
        X_test_pos_norm = X_test_pos_norm.transpose()

        print("After transpose")
        print(X_train_pos_norm.shape, y_train.shape)
        print(X_test_pos_norm.shape, y_test.shape)
        print("="*100)
```

```
Transpose of Pos score
After transpose
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

1.7.1.8 Neu Score

In [0]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['neu_score'].values.reshape(1,-1))

X_train_neu_norm = normalizer.transform(X_train['neu_score'].values.reshape(1,-1))
X_test_neu_norm = normalizer.transform(X_test['neu_score'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_neu_norm.shape, y_train.shape)
print(X_test_neu_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 33500) (33500,)
(1, 16500) (16500,)
====================================================================================================
```

In [0]:
```
print("Transpose of Neu score")

X_train_neu_norm = X_train_neu_norm.transpose()
X_test_neu_norm = X_test_neu_norm.transpose()

print("After transpose")
print(X_train_neu_norm.shape, y_train.shape)
print(X_test_neu_norm.shape, y_test.shape)
print("="*100)
```

```
Transpose of Neu score
After transpose
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

1.7.1.9 Compound Score

In [0]:
```
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['compound_score'].values.reshape(1,-1))

X_train_comp_norm = normalizer.transform(X_train['compound_score'].values.reshape(1,-1))
X_test_comp_norm = normalizer.transform(X_test['compound_score'].values.reshape(1,-1))

print("After vectorizations")
print(X_train_comp_norm.shape, y_train.shape)
print(X_test_comp_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(1, 33500) (33500,)
(1, 16500) (16500,)
====================================================================================================
```

In [0]:
```
print("Transpose of Compound score")

X_train_comp_norm = X_train_comp_norm.transpose()
X_test_comp_norm = X_test_comp_norm.transpose()

print("After transpose")
print(X_train_comp_norm.shape, y_train.shape)
print(X_test_comp_norm.shape, y_test.shape)
print("="*100)
```

```
Transpose of Compound score
After transpose
(33500, 1) (33500,)
(16500, 1) (16500,)
====================================================================================================
```

1.7.2 Categorical Data

Categorical Features for vectorization

```
1. Clean Categories
2. Clean Sub Categories
3. School State
4. Teacher Prefix
5. Project grade category
```

1.7.2.1 Clean Categories

In [0]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=True)
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_CC_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_test_CC_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_CC_ohe.shape, y_train.shape)
print(X_test_CC_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(33500, 9) (33500,)
(16500, 9) (16500,)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds', 'Health_Sports', 'Mat
h_Science', 'Literacy_Language']
====================================================================================================
```

1.7.2.2 Clean Sub Categories

In [0]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, binary=True)
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_CSC_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_test_CSC_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_CSC_ohe.shape, y_train.shape)
print(X_test_CSC_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(33500, 30) (33500,)
(16500, 30) (16500,)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular', 'Civics_Governmen
t', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Charac
terEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'E
arlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedScience
s', 'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
====================================================================================================
```

1.7.2.3 School State

In [0]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_dict.keys()), lowercase=False, binary=True)
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_test_state_ohe = vectorizer.transform(X_test['school_state'].values)

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

```
After vectorizations
(33500, 51) (33500,)
(16500, 51) (16500,)
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS', 'IA', 'ID', 'AR',
'CO', 'MN', 'OR', 'KY', 'MS', 'NV', 'MD', 'CT', 'TN', 'UT', 'AL', 'WI', 'VA', 'AZ', 'NJ', 'OK', 'WA', 'MA', 'LA',
'OH', 'MO', 'IN', 'PA', 'MI', 'SC', 'GA', 'IL', 'NC', 'FL', 'NY', 'TX', 'CA']
====================================================================================================
```

1.7.2.4 Teacher prefix

In [0]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict.keys()), lowercase=False, binary=True)
vectorizer.fit(X_train['clean_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['clean_teacher_prefix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['clean_teacher_prefix'].values)

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

```
After vectorizations
(33500, 5) (33500,)
(16500, 5) (16500,)
['Dr', 'Teacher', 'Mr', 'Ms', 'Mrs']
====================================================================================================
```

1.7.2.5 Project Grade category

In [0]:
```
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.keys()), lowercase=False, binary=Tr
ue)
vectorizer.fit(X_train['clean_project_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['clean_project_grade_category'].values)
X_test_grade_ohe = vectorizer.transform(X_test['clean_project_grade_category'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(33500, 4) (33500,)
(16500, 4) (16500,)
['9-12', '6-8', '3-5', 'PreK-2']
====================================================================================================
```

Assignment 11: TruncatedSVD

- step 1 Select the top 2k words from essay text and project_title (concatinate essay text with project title and then find the top 2k words) based on their `idf_` (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html) values
- step 2 Compute the co-occurance matrix with these 2k words, with window size=5 (ref (https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/))
- step 3 Use TruncatedSVD (http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html) on calculated co-occurance matrix and reduce its dimensions, choose the number of components (`n_components`) using elbow method (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/pca-code-example-using-non-visualization/)

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

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

2. TruncatedSVD

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

In [0]:
```
def extract_words(string):
    string = re.sub("\d+"," ",string)
    return re.sub("[^a-zA-Z]", ' ', string)

X_train["preprocessed_title_essays"] = X_train["preprocessed_title_essays"].apply(extract_words)
X_test["preprocessed_title_essays"] = X_test["preprocessed_title_essays"].apply(extract_words)
```

In [0]:
```
vectorizer_tfidf = TfidfVectorizer(min_df=1, stop_words='english')
vectorizer_tfidf.fit(X_train["preprocessed_title_essays"])

dictionary = dict(zip(vectorizer_tfidf.get_feature_names(), list(vectorizer_tfidf.idf_)))
tfidf_words = set(vectorizer_tfidf.get_feature_names())
```

In [0]:
```
top_words = np.array(sorted(zip(vectorizer_tfidf.get_feature_names(), list(vectorizer_tfidf.idf_)), key=lambda x:x[
1] )[:2000])
```

In [0]:
```
top_words
```

Out[0]:
```
array([['students', '1.0074002285135772'],
       ['school', '1.157644233270699'],
       ['learning', '1.347747166047801'],
       ...,
       ['factors', '6.030258838297401'],
       ['inspires', '6.030258838297401'],
       ['pretty', '6.030258838297401']], dtype='<U18')
```

2.2 Computing Co-occurance matrix

In [10]:
```
#Initialise empty Co-occurance matrix for sample data
cooccurance_matrix = pd.DataFrame(np.zeros((len(top_words),len(top_words))), index=top_words, columns=top_words)
cooccurance_matrix.head(10)
```

Out[10]:

	abc	pqr	def
abc	0.0	0.0	0.0
pqr	0.0	0.0	0.0
def	0.0	0.0	0.0

In [0]:
```
# Stores top k words
top_k_words = top_words[:,0]

def cooccurance_matrix_formation(sentence, window_size=5):
    """
    If window_size=2, it takes 2 words on right, 2 words on left of the current word.
    and updates the co-occurance matrix , while traversing that window till the end.
    """
    words = sentence.split()
    for index in range(len(words)):
        if words[index] in top_k_words:
          # Getting right side words
            left_index_start = index-window_size if (index-window_size)>=0 else 0
            for left_index in range(left_index_start, index):
                if (words[left_index] in top_k_words):
                    cooccurance_matrix[words[index]][words[left_index]]+=1

            # Getting right side words
            try:
                for right_index in range(index+1, index+window_size+1):
                    if (words[right_index] in top_k_words):
                        cooccurance_matrix[words[index]][words[right_index]]+=1
            except:
                pass
```

Calculating Cooccurence Matrix for Sample Data

In [4]:
```
import pandas as pd
import numpy as np

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

df = pd.DataFrame()
df['example_list'] = corpus
df.head()
```

Out[4]:

	example_list
0	abc def ijk pqr
1	pqr klm opq
2	lmn pqr xyz abc def pqr abc

In [0]:
```
top_words=['abc','pqr', 'def']
```

In [11]:
```
df['example_list'].apply(lambda x:cooccurance_matrix_formation(x,2))
```

Out[11]:
```
0    None
1    None
2    None
Name: example_list, dtype: object
```

In [12]:
```
cooccurance_matrix
```

Out[12]:

	abc	pqr	def
abc	0.0	3.0	3.0
pqr	3.0	0.0	2.0
def	3.0	2.0	0.0

In [0]:
```
#Initialise empty Co-occurance matrix
cooccurance_matrix = pd.DataFrame(np.zeros((top_words.shape[0],top_words.shape[0])), index=top_words[:,0], columns=
top_words[:,0])
cooccurance_matrix.head(10)
```

Out[0]:

	students	school	learning	classroom	learn	help	need	work	come	use	able	love	day	class	make	new	year	time	stud
students	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
school	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
learning	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
classroom	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
learn	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
help	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
need	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
work	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
come	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
use	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	

10 rows × 2000 columns

In [0]:
```
%%time
X_train['preprocessed_title_essays'].apply(lambda x:cooccurance_matrix_formation(x,5))
```

```
CPU times: user 54min 3s, sys: 793 ms, total: 54min 4s
Wall time: 54min 8s
```

Out[0]:
```
16875    None
18818    None
17086    None
40147    None
8224     None
         ...
21628    None
345      None
13870    None
25641    None
37981    None
Name: preprocessed_title_essays, Length: 33500, dtype: object
```

Co-occurance matrix of top_k_words

In [0]:
```
cooccurance_matrix
```

Out[0]:

	students	school	learning	classroom	learn	help	need	work	come	use	able	love	day	class
students	39210.0	40123.0	29251.0	25723.0	22757.0	19714.0	16923.0	14819.0	16244.0	13532.0	13610.0	13075.0	10765.0	10628.0
school	40123.0	17282.0	5355.0	4202.0	6475.0	2818.0	2855.0	3527.0	8615.0	1599.0	1524.0	3563.0	6494.0	2229.0
learning	29251.0	5355.0	6118.0	6920.0	3203.0	4807.0	3037.0	2394.0	1581.0	2915.0	2679.0	5404.0	2524.0	2051.0
classroom	25723.0	4202.0	6920.0	4386.0	3703.0	4387.0	3032.0	2594.0	1830.0	3468.0	2488.0	2393.0	2992.0	1249.0
learn	22757.0	6475.0	3203.0	3703.0	2692.0	3623.0	2580.0	2198.0	3819.0	1836.0	1877.0	4283.0	3814.0	1573.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
search	86.0	15.0	14.0	19.0	16.0	19.0	12.0	9.0	9.0	26.0	29.0	8.0	2.0	8.0
worlds	94.0	10.0	16.0	19.0	12.0	20.0	3.0	5.0	4.0	6.0	15.0	6.0	5.0	4.0
factors	128.0	39.0	20.0	9.0	17.0	4.0	6.0	12.0	7.0	5.0	8.0	9.0	2.0	3.0
inspires	85.0	31.0	51.0	26.0	19.0	8.0	4.0	21.0	9.0	3.0	6.0	15.0	22.0	5.0
pretty	86.0	49.0	12.0	25.0	10.0	8.0	9.0	17.0	12.0	5.0	2.0	14.0	6.0	11.0

2000 rows × 2000 columns

In [0]:
```
coocc_matrix=np.array(cooccurance_matrix)
```

In [0]:
```
coocc_matrix.sum(axis=1) , coocc_matrix.sum(axis=0), coocc_matrix.sum()
```

Out[0]:
```
(array([1.713832e+06, 5.405160e+05, 4.174530e+05, ..., 1.537000e+03,
        1.507000e+03, 1.368000e+03]),
 array([1.713832e+06, 5.405160e+05, 4.174530e+05, ..., 1.537000e+03,
        1.507000e+03, 1.368000e+03]),
 25829494.0)
```

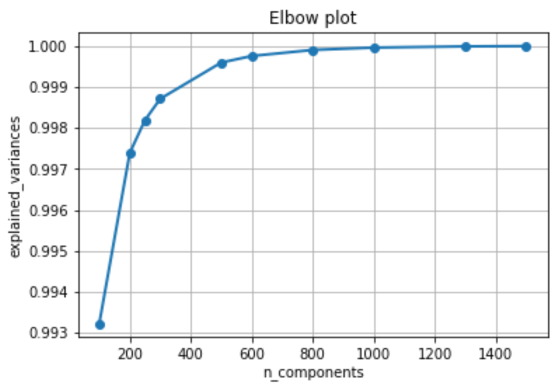
```
In [0]: import pickle
        with open("cooccurance_matrix_x_train.pkl", "wb") as f:
          pickle.dump(coocc_matrix, f)
```

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

```
In [0]: %%time
        from sklearn.decomposition import TruncatedSVD

        n_components = [100,200,250,300,500,600,800,1000,1300,1500]
        explained_variance_set=[]
        for components in tqdm(n_components):
            svd_model = TruncatedSVD(n_components=components, n_iter=7, random_state=28)
            svd_model.fit(coocc_matrix)
            explained_variance_set.append(svd_model.explained_variance_ratio_.sum())
            print(explained_variance_set)
```

```
 10%|█         | 1/10 [00:00<00:06,  1.30it/s]

[0.9932274725933956]

 20%|██        | 2/10 [00:02<00:07,  1.09it/s]

[0.9932274725933956, 0.9973900833923574]

 30%|███       | 3/10 [00:03<00:07,  1.13s/it]

[0.9932274725933956, 0.9973900833923574, 0.9981915869986305]

 40%|████      | 4/10 [00:05<00:08,  1.35s/it]

[0.9932274725933956, 0.9973900833923574, 0.9981915869986305, 0.9987104086359343]

 50%|█████     | 5/10 [00:08<00:09,  1.94s/it]

[0.9932274725933956, 0.9973900833923574, 0.9981915869986305, 0.9987104086359343, 0.9995949425971782]

 60%|██████    | 6/10 [00:13<00:10,  2.61s/it]

[0.9932274725933956, 0.9973900833923574, 0.9981915869986305, 0.9987104086359343, 0.9995949425971782, 0.99975462883
50833]

 70%|███████   | 7/10 [00:18<00:10,  3.57s/it]

[0.9932274725933956, 0.9973900833923574, 0.9981915869986305, 0.9987104086359343, 0.9995949425971782, 0.99975462883
50833, 0.9999024884902178]

 80%|████████  | 8/10 [00:26<00:09,  4.77s/it]

[0.9932274725933956, 0.9973900833923574, 0.9981915869986305, 0.9987104086359343, 0.9995949425971782, 0.99975462883
50833, 0.9999024884902178, 0.9999594673029059]

 90%|█████████ | 9/10 [00:37<00:06,  6.57s/it]

[0.9932274725933956, 0.9973900833923574, 0.9981915869986305, 0.9987104086359343, 0.9995949425971782, 0.99975462883
50833, 0.9999024884902178, 0.9999594673029059, 0.9999899949606782]

100%|██████████| 10/10 [00:49<00:00,  4.97s/it]

[0.9932274725933956, 0.9973900833923574, 0.9981915869986305, 0.9987104086359343, 0.9995949425971782, 0.99975462883
50833, 0.9999024884902178, 0.9999594673029059, 0.9999899949606782, 0.9999968406481884]
CPU times: user 1min 28s, sys: 9.21 s, total: 1min 37s
Wall time: 49.8 s
```

```
In [0]: plt.plot(n_components, explained_variance_set, marker='o', linewidth=2)
        plt.grid(True)
        plt.xlabel('n_components')
        plt.ylabel("explained_variances")
        plt.title("Elbow plot")
        plt.show()
```



```
In [0]: # Optimal n_components=500
        svd_best = TruncatedSVD(n_components=500, n_iter=7, random_state=28)
        svd_best.fit(coocc_matrix)
        svd_best.explained_variance_ratio_.sum()

Out[0]: 0.9995949425971782
```

In [0]:
```
svd_best.singular_values_
```

```
Out[0]: array([155231.52409959,  46345.2041992 ,  30748.91331676,  15493.02059708,
                14302.6482791 ,  11534.69480737,  11286.24742903,  10490.57643752,
                10013.66973026,   9470.78864825,   8737.89626976,   8590.84639609,
                 8363.61739545,   8072.12506546,   7531.84591144,   7236.67119314,
                 6215.09309522,   5975.72078983,   5935.82280135,   5802.89370541,
                 5718.87168224,   5635.17042416,   5147.79349115,   5096.38416531,
                 4942.90196851,   4905.36206987,   4832.80526559,   4816.12138571,
                 4762.64677049,   4721.61754486,   4630.90401022,   4335.52186891,
                 4238.57926736,   4236.35828027,   4074.63121615,   3993.81102388,
                 3889.75937226,   3840.57272849,   3797.83917878,   3773.06087318,
                 3684.52834515,   3582.82499893,   3514.67165907,   3461.89390721,
                 3386.90032861,   3261.61566261,   3224.8761441 ,   3171.34767634,
                 3050.49096627,   2945.95436078,   2937.25631016,   2826.41508236,
                 2802.87233994,   2704.39343497,   2651.63636023,   2642.81550753,
                 2538.84724505,   2516.99543638,   2474.72455101,   2462.02176628,
                 2411.07250132,   2369.19117505,   2351.46187307,   2330.19080095,
                 2318.4219468 ,   2255.19463431,   2224.57398077,   2213.97136789,
                 2173.66702669,   2172.52185854,   2113.12326192,   2067.0109519 ,
                 2037.66711266,   2012.6280587 ,   1984.89125989,   1975.78181182,
                 1974.36663819,   1927.3570288 ,   1912.05010417,   1876.49596207,
                 1874.36922115,   1848.75739446,   1809.40347159,   1808.49502521,
                 1789.39723115,   1781.85442638,   1764.21209192,   1704.35680984,
                 1691.95522419,   1680.09234006,   1659.12208822,   1643.81650665,
                 1614.48981684,   1593.76476553,   1579.84589651,   1570.33214205,
                 1554.30860882,   1550.98729434,   1532.66266844,   1511.22157616,
                 1498.43492782,   1493.2929068 ,   1490.03198461,   1475.95016346,
                 1460.52001043,   1431.67209357,   1425.50115646,   1403.75489031,
                 1398.03121476,   1391.33088572,   1358.19744999,   1357.40706691,
                 1354.55463166,   1326.93987829,   1314.26644393,   1305.3834537 ,
                 1302.47448852,   1280.53424085,   1271.08671116,   1258.03995964,
                 1242.71338493,   1236.27029672,   1228.49345532,   1217.2935614 ,
                 1206.29871239,   1205.52830492,   1184.57166108,   1173.29286629,
                 1169.45065344,   1164.55207211,   1152.14393322,   1149.92555443,
                 1135.21201733,   1128.69323833,   1120.10941371,   1114.4160216 ,
                 1109.29443198,   1104.61566066,   1099.62678461,   1096.40529728,
                 1081.50071986,   1075.11061774,   1056.74774055,   1048.74302523,
                 1044.6809583 ,   1044.35773469,   1036.52099021,   1018.65419817,
                 1010.44334191,   1006.04785996,   1002.98283144,    999.86738251,
                  979.57496029,    978.60704742,    975.66301164,    966.980683  ,
                  964.54086363,    959.65025629,    956.04637502,    944.12545701,
                  943.74858735,    935.23336272,    930.01635275,    922.08735421,
                  917.38105465,    903.73350114,    894.7194392 ,    892.34167012,
                  890.27139429,    884.03851221,    881.64862264,    871.47028177,
                  871.39103885,    862.60858567,    858.49461604,    854.71190976,
                  850.02762833,    841.13881553,    839.3793637 ,    831.45698148,
                  827.69705064,    819.28813965,    817.82977871,    813.41545299,
                  813.27094288,    805.59458969,    803.4051078 ,    798.32067759,
                  794.45653504,    793.51677218,    789.68152333,    779.82489851,
                  779.55552681,    775.67291152,    772.14764861,    770.87892448,
                  760.73119613,    760.04668319,    756.2845587 ,    750.75161921,
                  746.58928887,    745.61776186,    739.15082369,    736.67388866,
                  733.77169184,    728.70180246,    719.65789747,    719.44390562,
                  711.51329304,    708.5188469 ,    705.15928422,    702.7759093 ,
                  701.75749738,    694.17960956,    687.08093742,    687.06922849,
                  683.50023862,    680.96523597,    677.58541518,    674.06114086,
                  673.30264619,    667.03708842,    665.66473041,    662.42685182,
                  660.37375695,    657.40549508,    654.95047664,    651.35785886,
                  650.49538223,    644.48944143,    639.38932267,    639.10973678,
                  631.04769998,    630.07448837,    627.34533028,    625.35425861,
                  622.29288319,    610.75123098,    610.05570167,    609.63437597,
                  608.77462427,    607.80394097,    605.49540877,    602.35514561,
                  600.9999547 ,    597.04222328,    593.10105506,    593.03205898,
                  588.84966419,    588.15045095,    587.88650773,    584.19722004,
                  583.00063912,    577.0977154 ,    574.25669479,    572.56201643,
                  570.84736128,    569.27918878,    567.01030612,    562.7977541 ,
                  558.79143396,    557.9851576 ,    554.56327381,    552.13285915,
                  550.77233214,    549.88106548,    546.58563785,    544.65489802,
                  543.58272967,    542.39413793,    540.5047335 ,    535.9810668 ,
                  532.65473365,    532.12801871,    529.25664324,    527.07850503,
                  525.69307463,    524.60221268,    521.83599662,    521.09538389,
                  519.58613002,    519.05478576,    514.41499942,    513.74200197,
                  510.45910591,    508.86130242,    506.84754199,    504.57990798,
                  503.36922806,    501.84694466,    499.19415651,    498.87412252,
                  492.15099979,    488.61129552,    487.32665496,    487.20290247,
                  485.75087331,    485.33448064,    483.32655532,    480.63948588,
                  480.46083595,    478.46489628,    478.00932104,    473.45010184,
                  470.51509134,    468.79073089,    466.91608699,    464.8614251 ,
                  462.08781447,    461.79721348,    459.96315747,    459.46563664,
                  456.80238295,    456.39502652,    454.88686429,    454.04862031,
                  451.54239505,    449.14607923,    448.08464541,    447.85262737,
                  445.31606256,    441.00011657,    440.32458861,    439.53601746,
                  437.98302871,    435.37112514,    434.21665442,    431.12151099,
                  430.4997713 ,    427.76973202,    427.30457444,    426.51856259,
                  425.10903113,    421.38319264,    421.37987216,    417.52722238,
                  415.02854248,    413.91692669,    412.6093262 ,    409.66506991,
                  409.4810751 ,    407.5543779 ,    407.55327538,    405.13718388,
                  404.44632041,    403.08367471,    401.79085876,    399.33331165,
                  397.12485452,  395.82947124,  394.40769734,  389.96753472,
                  389.77506401,  388.11225536,  385.78722368,  385.5012638 ,
                  385.3835921 ,  384.13721612,  382.78576518,  380.26342294,
                  379.98800202,  378.05798081,  377.73052805,  376.11352528,
                  375.014892  ,  374.04914546,  371.00486603,  370.9003431 ,
                  370.5080522 ,  367.26778643,  366.82487202,  366.2220137 ,
                  364.57457621,  364.38632348,  361.45858566,  359.69196345,
                  358.06440495,  355.08992016,  355.07083182,  353.70483267,
                  352.60483446,  351.7641627 ,  349.70850002,  349.55479896,
                  346.40973846,  345.99279159,  345.74026028,  344.97811009,
                  344.19838803,  340.67191719,  340.5824817 ,  339.28807324,
                  338.00366881,  337.49629687,  335.89400619,  334.61736995,
                  333.63051918,  331.82345178,  331.31262972,  330.89583695,
                  329.96481305,  327.8447011 ,  327.44345092,  325.53500948,
                  324.06518885,  323.82472007,  322.23692401,  322.22395405,
                  320.69763377,  320.43973671,  319.52981197,  318.16160435,
                  316.02621605,  315.90001721,  314.87110355,  314.73322699,
                  313.3454249 ,  312.1958599 ,  311.03275474,  310.63836021,
                  308.98997865,  308.5932383 ,  307.41718566,  305.6068733 ,
                  303.32188961,  303.10876688,  302.43064969,  302.40323473,
                  299.5764244 ,  298.12521513,  297.94003904,  297.32307585,
                  294.74801967,  294.74104917,  294.1203123 ,  292.73957366,
                  292.49348937,  292.27403675,  291.24219036,  290.68618329,
                  288.12901782,  287.78958501,  286.58946647,  286.09011763,
                  285.78912304,  284.64558527,  282.4305895 ,  282.2994496 ,
                  281.92994392,  281.2180093 ,  279.57900698,  279.43109805,
                  277.57251692,  275.72205435,  275.29206955,  275.21123428,
                  274.78114929,  274.31061161,  273.02493025,  271.65870352,
                  270.37275602,  269.30430756,  268.6505084 ,  267.75687139,
                  267.18824803,  266.62388861,  265.29640781,  265.05760827,
                  264.53130168,  261.62179839,  261.27059805,  260.20393062,
                  258.88879209,  258.096953  ,  257.41856429,  256.92403117,
                  255.30429119,  254.99146895,  254.45952106,  253.53936014,
                  252.66643844,  251.18388792,  251.12642633,  249.37454163,
                  249.14835132,  248.51310009,  248.0901903 ,  247.62546835,
                  246.63938216,  245.3105845 ,  243.81850475,  243.46969572,
                  242.89679414,  242.4221703 ,  241.23626405,  240.51729601,
                  239.56349413,  238.92642916,  238.57455143,  237.44105859])
```

In [0]:
```
svd_best.components_.T.shape
```

Out[0]:
```
(2000, 500)
```

In [0]:
```
with open("svd_components_best.pkl", "wb") as f:
  pickle.dump(svd_best.components_, f)
```

Creating embeddings using optimal dimensions

In [0]:
```
def embeddings_optimal_dimensions(sentence, optimal_dim):
    vector = np.zeros(optimal_dim)
    word_count=0
    for word in sentence.split():
        try:
            index = cooccurance_matrix.index.get_loc(word)
            vector += svd.components_.T[index]
            word_count+=1
        except:
            # If word isn't present, do nothing
            pass
    if word_count!=0:
        vector /=word_count
    return vector
```

In [0]:
```
final_X_train = []
for sentence in tqdm(X_train["preprocessed_title_essays"]):
    final_X_train.append(embeddings_optimal_dimensions(sentence, 500))
```

```
100%|██████████| 33500/33500 [00:10<00:00, 3170.35it/s]
```

In [0]:
```
final_X_test = []
for sentence in tqdm(X_test["preprocessed_title_essays"]):
    final_X_test.append(embeddings_optimal_dimensions(sentence, 500))
```

```
100%|██████████| 16500/16500 [00:05<00:00, 3159.33it/s]
```

In [0]:
```
final_X_train = np.array(final_X_train)
final_X_test = np.array(final_X_test)
```

2.4 Merge the features from step 3 and step 4

In [0]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack, vstack

#set1 =  [categorical, numerical, project_title(BOW) , preprocessed_essay (BOW)]
X_train_final_set = hstack((X_train_CC_ohe,X_train_CSC_ohe,X_train_state_ohe,\
                       X_train_teacher_ohe,X_train_grade_ohe,X_train_TPPP_norm,\
                       X_train_price_norm,X_train_quantity_norm,X_train_esscnt_norm,\
                       X_train_titlecnt_norm,X_train_neg_norm,X_train_pos_norm,X_train_neu_norm,X_train_comp_norm,f
inal_X_train))

X_test_final_set = hstack((X_test_CC_ohe,X_test_CSC_ohe,X_test_state_ohe,X_test_teacher_ohe,\
                      X_test_grade_ohe,X_test_TPPP_norm,X_test_price_norm,\
                      X_test_quantity_norm,X_test_esscnt_norm,X_test_titlecnt_norm,X_test_neg_norm,\
                      X_test_pos_norm,X_test_neu_norm,X_test_comp_norm, final_X_test))



print("x_train {0} | y_train {1} ".format(X_train_final_set.shape , y_train.shape))
print("x_test  {0} | y_test  {1} ".format(X_test_final_set.shape , y_test.shape))
```

```
x_train (33500, 608) | y_train (33500,)
x_test  (16500, 608) | y_test  (16500,)
```

In [0]:
```
import pickle
with open("X_train_set.pkl", "wb") as f:
  pickle.dump(X_train_final_set, f)

with open("X_test_set.pkl", "wb") as f:
  pickle.dump(X_test_final_set, f)
```

In [0]:
```
import pickle
with open("y_train.pkl", "wb") as f:
  pickle.dump(y_train, f)

with open("y_test.pkl", "wb") as f:
  pickle.dump(y_test, f)
```

2.5 Apply XGBoost with Hyper Parameter Tuning Using GridSearch

In [0]:
```
X_train_final_set.shape
```

Out[0]:
```
(33500, 608)
```

In [0]:
```
import xgboost as xgb
from sklearn.metrics import confusion_matrix, roc_auc_score
from sklearn.model_selection import GridSearchCV
```

In [0]:
```
%%time
params = {"n_estimators":[10, 50, 100, 150, 200, 300, 500], "max_depth":[2,3,4,5]}

clf = xgb.XGBClassifier()

model = GridSearchCV(clf, param_grid=params, scoring="roc_auc" ,cv=3 , return_train_score=True)
model.fit(X_train_final_set, y_train)
```

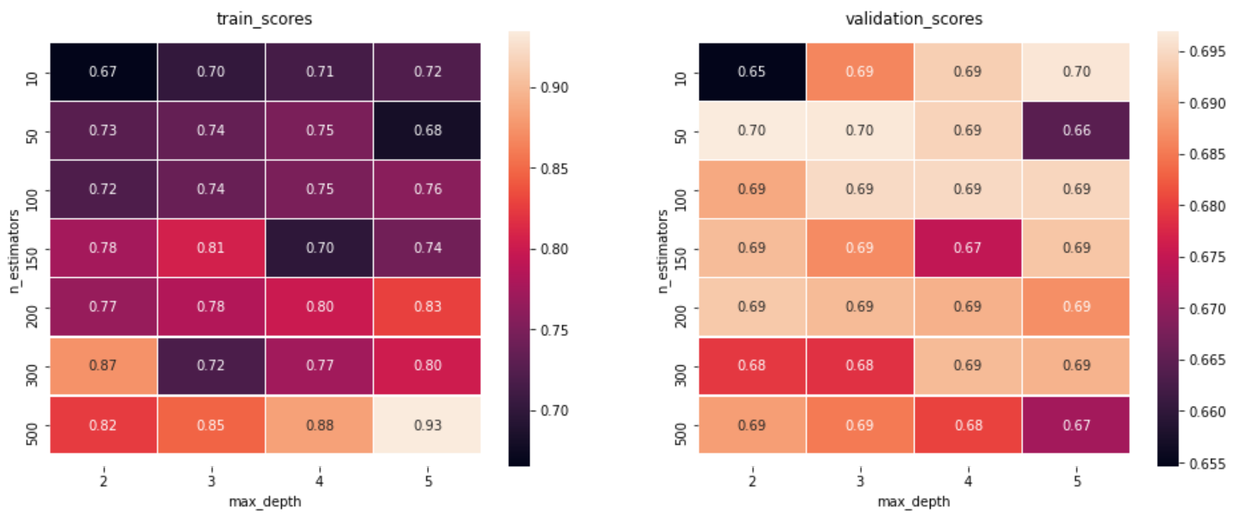
```
CPU times: user 4min 59s, sys: 1.02 s, total: 5min
Wall time: 5min 1s
```

In [0]:
```
print(model.best_score_)
print(model.best_params_)

print(model.cv_results_["mean_train_score"])
print(model.cv_results_["mean_test_score"])
```

```
0.6969305762533414
{'max_depth': 2, 'n_estimators': 200}
[0.66511789 0.7014349  0.71429376 0.72127822 0.72673012 0.73517713
 0.74844653 0.68084551 0.71998627 0.73694386 0.74862176 0.75794347
 0.77503006 0.80526241 0.69872738 0.74360537 0.76626776 0.78369951
 0.79897663 0.82783483 0.87388283 0.7181881  0.77293484 0.80184555
 0.8246972  0.8470977  0.88388042 0.93456297]
[0.65463062 0.68629467 0.69367518 0.69628762 0.69693058 0.69652559
 0.69427463 0.66450078 0.68977494 0.69494868 0.69485403 0.69451921
 0.69165149 0.68676857 0.67465541 0.69276582 0.69311042 0.69149122
 0.69057651 0.687125   0.67925079 0.67859722 0.69153406 0.69079561
 0.68852556 0.68512343 0.6803999  0.6717485 ]
```

Heatmap for finding best parameters

In [0]:
```
# plot a 3D plot (or) plot using heatmaps
fig, axs = plt.subplots(ncols=2, figsize=(16,6))
# http://seaborn.pydata.org/generated/seaborn.heatmap.html
t1= np.array(model.cv_results_["mean_train_score"]).reshape(len(params["n_estimators"]), len(params["max_depth"]))
sns.heatmap(t1 ,annot=True, ax=axs[0], yticklabels=params["n_estimators"], xticklabels=params["max_depth"], linewid
ths=0.3, fmt='0.2f')
axs[0].set_title("train_scores")
axs[0].set_ylabel("n_estimators")
axs[0].set_xlabel("max_depth")
axs[0].set_ylim(len(params["n_estimators"])+0.2, -0.2)

t2=np.array(model.cv_results_["mean_test_score"]).reshape(len(params["n_estimators"]), len(params["max_depth"]))
sns.heatmap(t2 ,annot=True, ax=axs[1], yticklabels=params["n_estimators"], xticklabels=params["max_depth"], linewid
ths=0.3, fmt='0.2f')
axs[1].set_title("validation_scores")
axs[1].set_ylabel("n_estimators")
axs[1].set_xlabel("max_depth")
axs[1].set_ylim(len(params["n_estimators"])+0.2, -0.2)
plt.show()
```



In [0]:

```
best_n_estimators = 200
best_max_depth = 2
xgb_model_best = xgb.XGBClassifier(n_estimators=best_n_estimators, max_depth=best_max_depth, random_state=28, njobs
=-1, verbosity=2)
xgb_model_best.fit(X_train_final_set, y_train)
```

```
[07:45:14] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:14] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:14] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:14] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:14] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
```

```
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:15] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
```

```
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:16] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
```

```
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
```

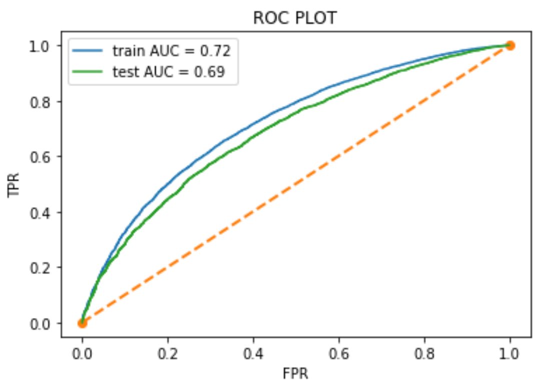
```
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:17] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:18] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:18] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:18] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:18] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:18] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:18] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
[07:45:18] INFO: /workspace/src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra nodes, 0 pruned node
s, max_depth=2
```

```
Out[0]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=1, gamma=0,
                      learning_rate=0.1, max_delta_step=0, max_depth=2,
                      min_child_weight=1, missing=None, n_estimators=200, n_jobs=1,
                      njobs=-1, nthread=None, objective='binary:logistic',
                      random_state=28, reg_alpha=0, reg_lambda=1, scale_pos_weight=1,
                      seed=None, silent=None, subsample=1, verbosity=2)
```

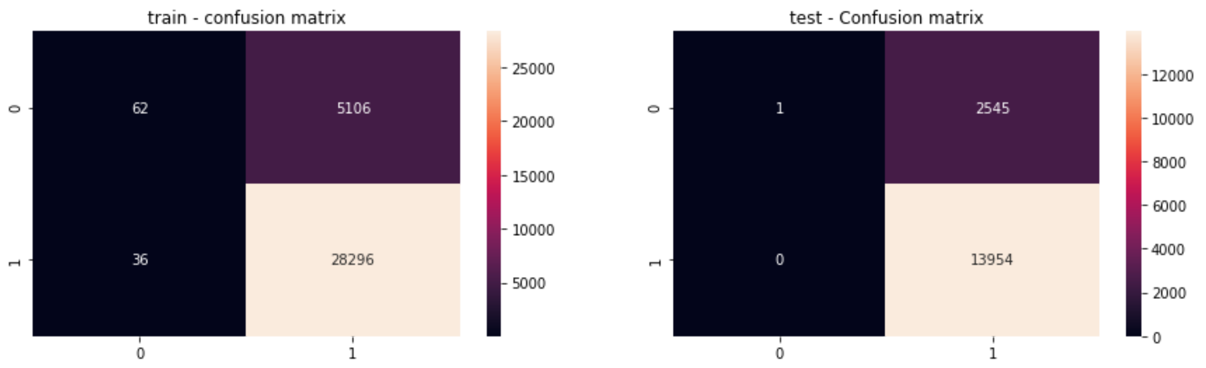
```
In [0]: import pickle
        with open("xgb_model_best.pkl", "wb") as f:
          pickle.dump(xgb_model_best, f)
```

```
In [0]: train_fpr, train_tpr, thresholds = roc_curve(y_train, xgb_model_best.predict_proba(X_train_final_set)[:,1])
        test_fpr, test_tpr, thresholds = roc_curve(y_test, xgb_model_best.predict_proba(X_test_final_set)[:,1])
```

In [0]:
```
plt.plot(train_fpr, train_tpr, label="train AUC = %0.2f"% auc(train_fpr, train_tpr))
#plt.plot(val_fpr, val_tpr, label="cv AUC = %0.2f"%auc(val_fpr, val_tpr))
plt.plot([0,1],[0,1], marker='o', linestyle='dashed', linewidth=2)
plt.plot(test_fpr, test_tpr, label="test AUC = %0.2f"%auc(test_fpr, test_tpr))
plt.legend()
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.title("ROC PLOT")
plt.show()
```



In [0]:
```
#https://stackoverflow.com/questions/38082602/plotting-multiple-different-plots-in-one-figure-using-seaborn
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#https://stackoverflow.com/questions/29647749/seaborn-showing-scientific-notation-in-heatmap-for-3-digit-numbers

fig, axs = plt.subplots(ncols=2, figsize=(15,4))
#train data
data = confusion_matrix(y_train, xgb_model_best.predict(X_train_final_set))
df_cm = pd.DataFrame(data, columns=[0,1], index = [0,1])
axs[0].set_title("train - confusion matrix")
sns.heatmap(df_cm, annot=True,ax=axs[0], fmt='d')

#test data
data = confusion_matrix(y_test, xgb_model_best.predict(X_test_final_set))
df_cm = pd.DataFrame(data, columns=[0,1], index = [0,1])
axs[1].set_title("test - Confusion matrix")
sns.heatmap(df_cm, annot=True,ax=axs[1], fmt='d')

plt.show()
```



3. Conclusion

```
1. We have considered only 50k data and later split the Data into 33k, 16k.
2. We have done vectorization of Categorical features using one-hot encoding, and standardization of numerical features.
3. First we calculated IDF for all words in essays + project_titles corpus and extracted top 2000 words basaed on IDF sc
ores.
4. Then we build Co-occurance matirx from essays + project_titles corpus with 5 as context-window size.
5. We have applied TSVD on Coocurance matrix and reduced the dimensionality, preserving max variance by optimal componen
ts.
5. By using Elbow plot, we observed that optimal components is 500.
6. Then we formed a final matrix, by converting every sentence into 500 dimensions and stacked all categorical features,
numerical features with final matirx to form final train and test datasets.
7. Then we have performed hyper parameter tuning using GridSearch() and applied XGBoost model on train data.
8. Finally we have plotted ROC-AUC curve and constructed confusion matrix on both train and test data.
```