11_DonorsChoose_TruncatedSVD

April 5, 2020

1 DonorsChoose

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

```
Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result

How to scale current manual processes and resources to screen 500,000 projects so that they ca
<li>How to increase the consistency of project vetting across different volunteers to improve
<li>How to focus volunteer time on the applications that need the most assistance</li>
</ul>
```

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.

1.1 About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`

`project_title` | Title of the project. **Examples:**
Art Will Make You Happy!
First Grade Fun
`project_grade_category` | Grade level of students for which the project is targeted. One of the following enumerated values:
Grades PreK-2
Grades 3-5
Grades 6-8
Grades 9-12
`project_subject_categories` | One or more (comma-separated) subject categories for the project from the following enumerated list of values:
Applied Learning

Care & Hunger
Health & Sports
History & Civics
Literacy & Language
Math & Science
Music & The Arts
Special Needs
Warmth
Examples:
Music & The Arts
Literacy & Language, Math & Science

`school_state` | State where school is located (Two-letter U.S. postal code). **Example:** `WY`

`project_subject_subcategories` | One or more (comma-separated) subject subcategories for the project. **Examples:**

Literacy
Literature & Writing, Social Sciences

`project_resource_summary` | An explanation of the resources needed for the project. **Example:**

My students need hands on literacy materials to manage sensory needs!

`project_essay_1` | First application essay
`project_essay_2` | *Second application essay* `project_essay_3` | Third application essay
`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`

Feature	Description
`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.

1.1.1 Notes on the Essay Data

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

project_essay_1: "Introduce us to your classroom"

project_essay_2: "Tell us more about your students"

project_essay_3: "Describe how your students will use the materials you're requesting"

project_essay_3: "Close by sharing why your project will make a difference"

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

project_essay_1: "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."

project_essay_2: "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

```
In [0]: !pip install chart_studio
```

```
Requirement already satisfied: chart_studio in /usr/local/lib/python3.6/dist-packages (1.0.0)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: plotly in /usr/local/lib/python3.6/dist-packages (from chart_stu
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from chart_studio
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (from chart_s
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/dist-packages (from r
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/python3.6/dist-packages
```

In [0]:
```
import chart_studio.plotly as py
import plotly.graph_objs as go
```

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

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

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

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

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

from tqdm import tqdm
import os

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

1.2 1.1 Reading Data

```
In [0]: 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=947318989803-6

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

```
In [0]: import pandas
        #data = pandas.read_csv(r'C:\Users\ASUS\Downloads\Applied AI\Assignments - Applied AI\
        # we are taking 50000 data points in project data
        project_data = pd.read_csv('/content/drive/My Drive/Assignments_DonorsChoose_2018/train
        resource_data = pd.read_csv('/content/drive/My Drive/Assignments_DonorsChoose_2018/reso
```

```
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
```

1.3 1.2 preprocessing of `project_subject_categories`

```
In [0]: catogories = list(project_data['project_subject_categories'].values)
        # remove special characters from list of strings python: https://stackoverflow.com/a/4
```

```
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-str
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-pyt
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", "Warmt
        if 'The' in j.split(): # this will split each of the catogory based on space "
            j=j.replace('The','') # if we have the words "The" we are going to replace
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing sp
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.4 1.3 preprocessing of project_subject_subcategories

```
In [0]: sub_catogories = list(project_data['project_subject_subcategories'].values)
        # remove special characters from list of strings python: https://stackoverflow.com/a/4

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

        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", "Warmt
                if 'The' in j.split(): # this will split each of the catogory based on space "
                    j=j.replace('The','') # if we have the words "The" we are going to replace
                j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"
                temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing sp
                temp = temp.replace('&','_')
            sub_cat_list.append(temp.strip())

        project_data['clean_subcategories'] = sub_cat_list
```

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

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/408403
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]))
```

Preprocessing of project_grade_category

```
In [0]: project_grade = list(project_data['project_grade_category'].values)
        # remove special characters from list of strings python: https://stackoverflow.com/a/4

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

        grade_cat_list = []
        for i in project_grade:
            # consider we have text like this:
            for j in i.split(' '): #    # split by spae
                j=j.replace('Grades','')# clean grades from the row
            grade_cat_list.append(j.strip())



        project_data['grade_cat_list'] = grade_cat_list
        project_data.drop(['project_grade_category'], axis=1, inplace=True)
```

1.5 1.3 Text preprocessing

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

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

```
Out[0]:    Unnamed: 0  ...                                              essay
        0      160221  ...  My students are English learners that are work...
        1      140945  ...  Our students arrive to our school eager to lea...

        [2 rows x 18 columns]
```

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

In [0]:
```
# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[2000])
print("="*50)
print(project_data['essay'].values[999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love learning, a
==================================================
How do you remember your days of school? Was it in a sterile environment with plain walls, rows
==================================================
Describing my students isn't an easy task.  Many would say that they are inspirational, creativ
==================================================
Welcome to our spectacular 1st and 2nd grade ELL classroom.  I have the most amazing class of m
==================================================
```

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[2000])
print(sent)
print("="*50)
```

```
Describing my students is not an easy task.  Many would say that they are inspirational, creati
==================================================
```

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

```
Describing my students is not an easy task.  Many would say that they are inspirational, creati
```

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

```
Describing my students is not an easy task Many would say that they are inspirational creative
```

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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', '
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "th
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'o
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'no
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't"
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'migh
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'w
            '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, 1745.65it/s]
```

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

        project_data.head(2)
```

```
Out[0]:    Unnamed: 0  ...                                preprocessed_essays
        0      160221  ...  my students english learners working english s...
        1      140945  ...  our students arrive school eager learn they po...

        [2 rows x 19 columns]
```

1.4 Preprocessing of project_title

```
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['project_title'].values[2000])
        print(sent)
        print("="*50)
```

```
Steady Stools for Active Learning
==================================================
```

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

```
Steady Stools for Active Learning
```

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

Steady Stools for Active Learning

```
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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him',
                    'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', '
                    'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "th
                    'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
                    'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as
                    'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through
                    'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'ov
                    'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any
                    'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too
                    's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'no
                    've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't",
                    "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'migh
                    "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'w
                    'won', "won't", 'wouldn', "wouldn't"]
```

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

```
100%|| 109248/109248 [00:02<00:00, 41973.87it/s]
```

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

        project_data.head(2)
```

```
Out[0]:    Unnamed: 0  ...                          preprocessed_titles
        0      160221  ...  educational support english learners home
        1      140945  ...           wanted projector hungry learners

        [2 rows x 20 columns]
```

Join train & Resource dataset

```
In [0]: price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_
        project_data = pd.merge(project_data, price_data, on='id', how='left')
        project_data.head(2)

Out[0]:    Unnamed: 0       id  ...  price quantity
        0      160221  p253737  ...  154.6       23
        1      140945  p258326  ...  299.0        1

        [2 rows x 22 columns]
```

Train Test split

```
In [0]: from sklearn.utils import resample
        p_d = resample(project_data,n_samples = 20000)

In [0]: y = p_d["project_is_approved"]
        X = p_d.drop("project_is_approved",axis = 1)

In [0]: # train test split
        from sklearn.model_selection import train_test_split
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y)

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

100%|| 16000/16000 [00:09<00:00, 1751.41it/s]


In [0]: # Combining all the above stundents
        from tqdm import tqdm
        test_preprocessed_essays = []
        # tqdm is for printing the status bar
        for sentance in tqdm(X_test['essay'].values):
            sent = decontracted(sentance)
            sent = sent.replace('\\r', ' ')
            sent = sent.replace('\\"', ' ')
            sent = sent.replace('\\n', ' ')
```

```
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        # https://gist.github.com/sebleier/554280
        sent = ' '.join(e for e in sent.split() if e.lower() not in stopwords)
        test_preprocessed_essays.append(sent.lower().strip())
    X_test["essays"] = test_preprocessed_essays
```

```
100%|| 4000/4000 [00:02<00:00, 1752.91it/s]
```

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

```
Out[0]: 'my kindergarten students varied disabilities ranging speech language delays cognitive
```

1.4 Preprocessing of project_title

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

```
100%|| 16000/16000 [00:00<00:00, 38295.84it/s]
```

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

```
100%|| 4000/4000 [00:00<00:00, 38436.92it/s]
```

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

```
Out[0]: 'steady stools active learning'
```

```
In [0]: # concatening project_essay + project_title
        X_train["concat"] = X_train["essays"] + X_train["project_titles"]
        X_test["concat"] = X_test["essays"] + X_test["project_titles"]
```

1.6 1.5 Preparing data for models

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

```
Out[0]: Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
               'project_submitted_datetime', 'project_title', 'project_essay_1',
               'project_essay_2', 'project_essay_3', 'project_essay_4',
               'project_resource_summary',
               'teacher_number_of_previously_posted_projects', 'project_is_approved',
               'clean_categories', 'clean_subcategories', 'grade_cat_list', 'essay',
               'preprocessed_essays', 'preprocessed_titles', 'price', 'quantity'],
              dtype='object')
```

we are going to consider

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

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

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

1.6.1 1.5.1 Vectorizing Categorical data

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

```
In [0]: # we use count vectorizer to convert the values into one
        from sklearn.feature_extraction.text import CountVectorizer
        vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False,
        categories_one_hot = vectorizer.fit_transform(project_data['clean_categories'].values)
```

```
        print(vectorizer.get_feature_names())
        print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
        print("="*100)
        cat_features = vectorizer.get_feature_names()
        print(cat_features)
        print(len(cat_features))
        print("="*100)
        X_train_cl_categories_ohe = vectorizer.transform(X_train['clean_categories'].values)
        X_test_cl_categories_ohe = vectorizer.transform(X_test['clean_categories'].values)
```

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

```
In [0]: # we use count vectorizer to convert the values into one
        vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fa
        sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].
        print(vectorizer.get_feature_names())
        print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
        print("="*100)
        sub_cat_features=vectorizer.get_feature_names()
        print(sub_cat_features)
        print(len(sub_cat_features))
        print("="*100)
        X_train_cl_subcategories_ohe = vectorizer.transform(X_train['clean_subcategories'].valu
        X_test_cl_subcategories_ohe = vectorizer.transform(X_test['clean_subcategories'].values
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
Shape of matrix after one hot encodig  (109248, 30)
====================================================================================================
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
30
====================================================================================================
```

```
In [0]: # you can do the similar thing with state, teacher_prefix and project_grade_category a
```

```
In [0]: # count of all the words in corpus python: https://stackoverflow.com/a/22898595/408403
        my_counter = Counter()
        for word in project_data['school_state'].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]:
```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fal
categories_one_hot = vectorizer.fit_transform(project_data['school_state'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
print("="*100)
state_features=vectorizer.get_feature_names()
print(state_features)
print(len(state_features))
print("="*100)
X_train_school_state_ohe = vectorizer.transform(X_train['school_state'].values)
X_test_school_state_ohe = vectorizer.transform(X_test['school_state'].values)
```

```
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS
Shape of matrix after one hot encodig  (109248, 51)
====================================================================================================
['VT', 'WY', 'ND', 'MT', 'RI', 'SD', 'NE', 'DE', 'AK', 'NH', 'WV', 'ME', 'HI', 'DC', 'NM', 'KS
51
====================================================================================================
```

In [0]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/408403
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna(" ")
my_counter = Counter()
for word in project_data['teacher_prefix'].values.astype('str'):  #https://stackoverfl
    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]:
```
# we use count vectorizer to convert the values into one
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fal
prefix_one_hot = vectorizer.fit_transform(project_data['teacher_prefix'].values.astype
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",prefix_one_hot.shape)
print("="*100)
prefix_features=vectorizer.get_feature_names()
print(prefix_features)
print(len(prefix_features))
print("="*100)
X_train_teacher_prefix_ohe = vectorizer.transform(X_train['teacher_prefix'].values.asty
X_test_teacher_prefix_ohe = vectorizer.transform(X_test['teacher_prefix'].values.astype
```

```
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
Shape of matrix after one hot encodig  (109248, 5)
====================================================================================================
['Dr.', 'Teacher', 'Mr.', 'Ms.', 'Mrs.']
```

```
5
==========================================================================================
```

```
In [0]: # count of all the words in corpus python: https://stackoverflow.com/a/22898595/408403
        my_counter = Counter()
        for word in project_data['grade_cat_list'].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]: # we use count vectorizer to convert the values into one
        from sklearn.feature_extraction.text import CountVectorizer
        vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=Fa
        categories_one_hot = vectorizer.fit_transform(project_data['grade_cat_list'].values)
        print(vectorizer.get_feature_names())
        print("Shape of matrix after one hot encodig ",categories_one_hot.shape)
        print("="*100)
        grade_features=vectorizer.get_feature_names()
        print(grade_features)
        print(len(grade_features))
        print("="*100)
        X_train_grade_ohe = vectorizer.transform(X_train['grade_cat_list'].values)
        X_test_grade_ohe = vectorizer.transform(X_test['grade_cat_list'].values)
```

```
['9-12', '6-8', '3-5', 'PreK-2']
Shape of matrix after one hot encodig  (109248, 4)
==========================================================================================
['9-12', '6-8', '3-5', 'PreK-2']
4
==========================================================================================
```

1.6.2 1.5.2 Vectorizing Text data

1.5.2.2 TFIDF vectorizer

```
In [0]: from sklearn.feature_extraction.text import TfidfVectorizer
        vectorizer = TfidfVectorizer(min_df=10, ngram_range=(1,4), max_features=10000)
        text_tfidf = vectorizer.fit(X_train["concat"].values)
        text_tfidf = text_tfidf.fit_transform(X_train["concat"].values)
        essays_features_tf=vectorizer.get_feature_names()
        #print(essays_features)
        print(len(essays_features_tf))
```

```
10000
```

Getting top 2k features according to idf_ values

```
In [0]: #https://kavita-ganesan.com/extracting-keywords-from-text-tfidf/#.XlbJtKgzbIU
        #https://stackoverflow.com/questions/52972368/select-top-n-tfidf-features-for-a-given-

In [0]: #concatinating features
        n=2000
        text_tfidf_top = [np.argsort(text_tfidf[i: i+500, :].toarray(), axis=1)[:, :n]
                  for i in range(0, text_tfidf.shape[0], 500)]

In [0]: np.concatenate(text_tfidf_top, axis=0).shape

Out[0]: (16000, 2000)

In [0]: myDict = {}
        myDict['index'] = range(len(vectorizer.idf_))
        myDict['idf'] = vectorizer.idf_

In [0]: df = pd.DataFrame(myDict)
        df.sort_values(['idf'], axis=0,
                       ascending=True, inplace=True)
        words = vectorizer.get_feature_names()
        top_words = []
        for i in df['index'].values:
            top_words.append(words[i])

        top_words = top_words[0:2000]
        print(len(top_words))

2000


In [0]: def get_corpus(df):
            """
            This function returns list of all words in corpus.

            """
            corpus = " "
            for ew in df["concat"].values:
                corpus += ew
            return tuple(corpus.split())

In [0]: train_corpus = get_corpus(X_train)
        print("total number of words in train corpus ",len(train_corpus))

        test_corpus = get_corpus(X_test)
        print("total number of words in test corpus ",len(test_corpus))

total number of words in train corpus  2237605
total number of words in test corpus  559664
```

1.6.3 1.5.3 Vectorizing Numerical features

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

        # price_standardized = standardScalar.fit(project_data['price'].values)
        # this will rise the error
        # ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ...
        # Reshape your data either using array.reshape(-1, 1)

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

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

```
Mean : 298.1193425966608, Standard deviation : 367.49634838483496
```

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

```
Out[0]: array([[-0.3905327 ],
               [ 0.00239637],
               [ 0.59519138],
               ...,
               [-0.15825829],
               [-0.61243967],
               [-0.51216657]])
```

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

        # price_standardized = standardScalar.fit(project_data['price'].values)
        # this will rise the error
        # ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.   ...
        # Reshape your data either using array.reshape(-1, 1)

        quantity_scalar = StandardScaler()
        quantity_scalar.fit(project_data['quantity'].values.reshape(-1,1)) # finding the mean
        print(f"Mean : {quantity_scalar.mean_[0]}, Standard deviation : {np.sqrt(quantity_scala

        # Now standardize the data with above maen and variance.
        quantity_standardized = quantity_scalar.transform(project_data['quantity'].values.resha
```

```
        tr_quantity_standardized = quantity_scalar.transform(X_train['quantity'].values.reshape
        #cv_quantity_standardized = quantity_scalar.transform(X_cv['quantity'].values.reshape(
        te_quantity_standardized = quantity_scalar.transform(X_test['quantity'].values.reshape
```

```
Mean : 16.965610354422964, Standard deviation : 26.182821919093175
```

```
In [0]: quantity_standardized
```

```
Out[0]: array([[ 0.23047132],
               [-0.60977424],
               [ 0.19227834],
               ...,
               [-0.4951953 ],
               [-0.03687954],
               [-0.45700232]])
```

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

        # price_standardized = standardScalar.fit(project_data['price'].values)
        # this will rise the error
        # ValueError: Expected 2D array, got 1D array instead: array=[725.05 213.03 329.    ...
        # Reshape your data either using array.reshape(-1, 1)

        teacher_number_of_previously_posted_projects_scalar = StandardScaler()
        teacher_number_of_previously_posted_projects_scalar.fit(project_data['teacher_number_o
        print(f"Mean : {teacher_number_of_previously_posted_projects_scalar.mean_[0]}, Standar

        # Now standardize the data with above mean and variance.
        teacher_number_of_previously_posted_projects_standardized = teacher_number_of_previous
        tr_teacher_ppp_standardized = teacher_number_of_previously_posted_projects_scalar.trans
        #cv_teacher_ppp_standardized = teacher_number_of_previously_posted_projects_scalar.tra
        te_teacher_ppp_standardized = teacher_number_of_previously_posted_projects_scalar.trans
```

```
Mean : 11.153165275336848, Standard deviation : 27.77702641477403
```

__ Computing Sentiment Scores__

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

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

        sid = SentimentIntensityAnalyzer()
```

```
for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest
for learning my students learn in many different ways using all of our senses and multi
of techniques to help all my students succeed students in my class come from a variety
for wonderful sharing of experiences and cultures including native americans our school
learners which can be seen through collaborative student project based learning in and
in my class love to work with hands on materials and have many different opportunities
mastered having the social skills to work cooperatively with friends is a crucial aspe
montana is the perfect place to learn about agriculture and nutrition my students love
in the early childhood classroom i have had several kids ask me can we try cooking with
and create common core cooking lessons where we learn important math and writing concep
food for snack time my students will have a grounded appreciation for the work that wen
of where the ingredients came from as well as how it is healthy for their bodies this p
nutrition and agricultural cooking recipes by having us peel our own apples to make hom
and mix up healthy plants from our classroom garden in the spring we will also create e
shared with families students will gain math and literature skills as well as a life le
nannan'
ss = sid.polarity_scores(for_sentiment)

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

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

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

The twython library has not been installed. Some functionality from the twitter package will no
```

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

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_` values
- step 2 Compute the co-occurance matrix with these 2k words, with window size=5 (ref)
- step 3 Use TruncatedSVD on calculated co-occurance matrix and reduce its dimensions, choose the number of components (`n_components`) using elbow method >- The shape of the matrix after TruncatedSVD will be 2000*n, i.e. each row represents a vector form of the corresponding word. >- Vectorize the essay text and project titles using these word vectors. (while vectorizing, do ignore all the words which are not in top 2k words)
- step 4 Concatenate these truncatedSVD matrix, with the matrix with features
  school_state : categorical data
  clean_categories : categorical data
  clean_subcategories : categorical data
  project_grade_category :categorical data
  teacher_prefix : categorical data

quantity : numerical data
teacher_number_of_previously_posted_projects : numerical data
price : numerical data
sentiment score's of each of the essay : numerical data
number of words in the title : numerical data
number of words in the combine essays : numerical data
word vectors calculated in step 3 : numerical data

- step 5: Apply GBDT on matrix that was formed in step 4 of this assignment, DO REFER THIS BLOG: XGBOOST DMATRIX
  step 6:Hyper parameter tuning (Consider any two hyper parameters)
  Find the best hyper parameter which will give the maximum AUC value
  Find the best hyper paramter using k-fold cross validation or simple cross validation data
  Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

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

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

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

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

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

    def predict(self, X):
        num2label = {i: label for label, i in self.label2num.items()}
        Y = self.predict_proba(X)
        y = np.argmax(Y, axis=1)
        return np.array([num2label[i] for i in y])

    def predict_proba(self, X):
        dtest = xgb.DMatrix(X)
```

```
            return self.clf.predict(dtest)

        def score(self, X, y):
            Y = self.predict_proba(X)[:,1]
            return roc_auc_score(y, Y)

        def get_params(self, deep=True):
            return self.params

        def set_params(self, **params):
            if 'num_boost_round' in params:
                self.num_boost_round = params.pop('num_boost_round')
            if 'objective' in params:
                del params['objective']
            self.params.update(params)
            return self


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

    clf = GridSearchCV(clf, parameters)
    X = np.array([[1,2], [3,4], [2,1], [4,3], [1,0], [4,5]])
    Y = np.array([0, 1, 0, 1, 0, 1])
    clf.fit(X, Y)

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

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

2. TruncatedSVD

2.2 Computing Co-occurance matrix

```
In [0]: # https://stackoverflow.com/a/41663359/9371069 - Covariance Matrix

In [0]: vcv = np.zeros([2000,2000])

In [0]: def CovarMatrix(feat_dict, top_2000 ):
            #cm = np.array(c)
            #vcv = np.zeros([2000,2000])
            for eachlength in range(len(feat_dict)):
                word = feat_dict[eachlength]
                for window in range(1,6):
                    #print(eachlength + window)
                    #print(word,feat_dict[eachlength + window ])
                    #print(word in top_2000,feat_dict[eachlength + window ] in top_2000)
                    if(eachlength + window < len(feat_dict)):
                        if (word in top_2000 and feat_dict[eachlength + window] in top_2000):
                            i = top_2000.index(word)
                            j = top_2000.index(feat_dict[eachlength + window] )
                            vcv[i][j] = vcv[i][j] + 1
                            #print(i,j,word,feat_dict[eachlength + window])

                    if(word in top_2000 and eachlength - window >= 0):
                        if (feat_dict[eachlength - window ] in top_2000):
                            i = top_2000.index(word)
                            j = top_2000.index(feat_dict[eachlength - window ] )
                            vcv[i][j] = vcv[i][j] + 1

            return vcv

In [0]: for sentence in tqdm(train_preprocessed_essays):
            CovarMatrix(sentence.split(), top_words)

        for sentence in tqdm(train_preprocessed_titles):
            CovarMatrix(sentence.split(), top_words)

100%|| 16000/16000 [08:24<00:00, 31.69it/s]
100%|| 16000/16000 [00:07<00:00, 2173.10it/s]


In [0]: print (vcv)

[[2.3310e+04 2.0648e+04 1.4974e+04 ... 0.0000e+00 8.6000e+01 0.0000e+00]
 [2.0648e+04 9.2720e+03 2.8030e+03 ... 0.0000e+00 2.7000e+01 0.0000e+00]
 [1.4974e+04 2.8030e+03 3.1320e+03 ... 0.0000e+00 1.8000e+01 0.0000e+00]
 ...
 [0.0000e+00 0.0000e+00 0.0000e+00 ... 0.0000e+00 0.0000e+00 0.0000e+00]
```

```
 [8.6000e+01 2.7000e+01 1.8000e+01 ... 0.0000e+00 0.0000e+00 0.0000e+00]
 [0.0000e+00 0.0000e+00 0.0000e+00 ... 0.0000e+00 0.0000e+00 0.0000e+00]]
```

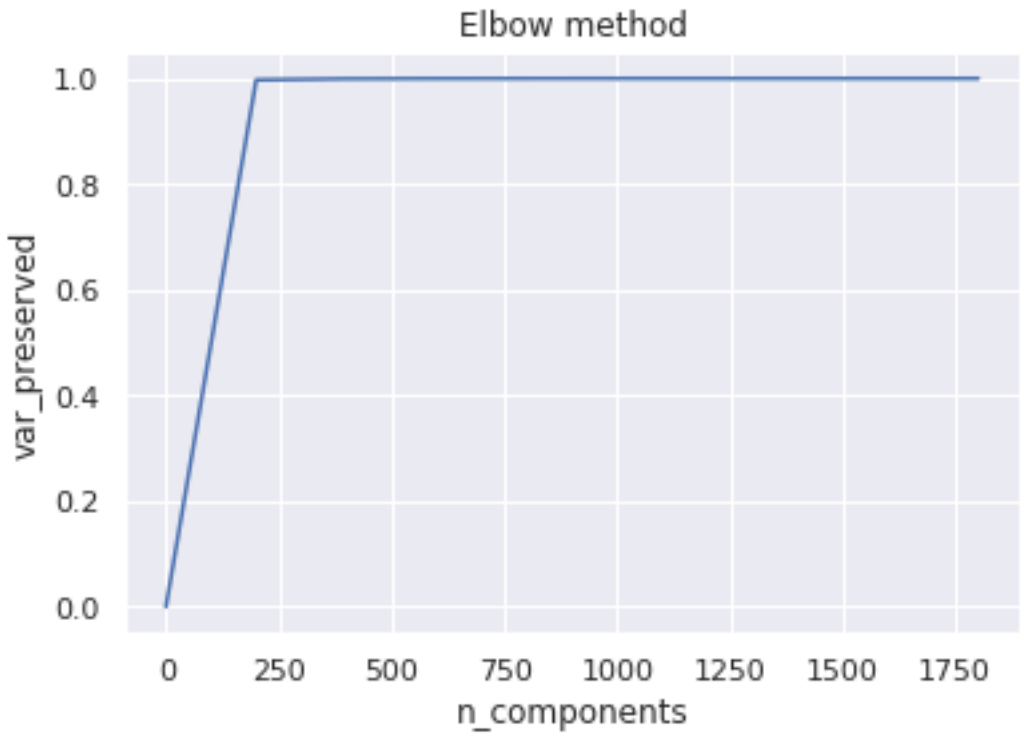
2.3 Applying TruncatedSVD and Calculating Vectors for `essay` and `project_title`

In [0]:
```
'''
from sklearn.decomposition import TruncatedSVD
pca = TruncatedSVD()
pca.n_components = 1999
pca_data = pca.fit(vcv)
'''
```

In [0]:
```
'''
percentage_var_explained = pca.explained_variance_ / np.sum(pca.explained_variance_);
cum_var_explained = np.cumsum(percentage_var_explained)
'''
```

In [0]:
```
#https://chrisalbon.com/machine_learning/feature_engineering/select_best_number_of_com
```

In [0]:
```
from sklearn.decomposition import TruncatedSVD
n_comp = list(range(0,2000,200))
var_preserved = []
for i in n_comp:
    tsvd = TruncatedSVD(n_components=i)
    tsvd.fit(vcv)
    var_preserved.append(tsvd.explained_variance_ratio_.sum())

sns.set()
plt.plot(n_comp,var_preserved)
plt.ylabel("var_preserved")
plt.xlabel("n_components")
plt.title("Elbow method")
plt.show()
```




```
In [0]: svd = TruncatedSVD(n_components = 220)

        truncated_svd = svd.fit_transform(vcv)
```

2.4 Merge the features from step 3 and step 4

```
In [0]: # average Word2Vec
        # compute average word2vec for each review.
        avg_w2v_vectors_tr = []; # the avg-w2v for each sentence/review is stored in this list
        for sentence in tqdm(train_preprocessed_essays): # for each review/sentence
            vector = np.zeros(220) # as word vectors are of zero length
            cnt_words =0; # num of words with a valid vector in the sentence/review
            for word in sentence.split(): # for each word in a review/sentence
                if word in top_words:
                    i = top_words.index(word)
                    vector += truncated_svd[i]
                    cnt_words += 1
            if cnt_words != 0:
                vector /= cnt_words
            avg_w2v_vectors_tr.append(vector)

        print(len(avg_w2v_vectors_tr))
        print(len(avg_w2v_vectors_tr[0]))
```

```
100%|| 16000/16000 [00:32<00:00, 486.44it/s]

16000
220
```

```
In [0]: # average Word2Vec
        # compute average word2vec for each review.
        avg_w2v_vectors_te = []; # the avg-w2v for each sentence/review is stored in this list
        for sentence in tqdm(test_preprocessed_essays): # for each review/sentence
            vector = np.zeros(220) # as word vectors are of zero length
            cnt_words =0; # num of words with a valid vector in the sentence/review
            for word in sentence.split(): # for each word in a review/sentence
                if word in top_words:
                    i = top_words.index(word)
                    vector += truncated_svd[i]
                    cnt_words += 1
            if cnt_words != 0:
                vector /= cnt_words
            avg_w2v_vectors_te.append(vector)

        print(len(avg_w2v_vectors_te))
        print(len(avg_w2v_vectors_te[0]))
```

```
100%|| 4000/4000 [00:08<00:00, 485.66it/s]

4000
220
```

```
In [0]: # average Word2Vec
        # compute average word2vec for each review.
        avg_w2v_vectors_preprocessed_project_title_tr = []; # the avg-w2v for each sentence/re
        for sentence in tqdm(train_preprocessed_titles): # for each review/sentence
            vector = np.zeros(220) # as word vectors are of zero length
            cnt_words =0; # num of words with a valid vector in the sentence/review
            for word in sentence.split(): # for each word in a review/sentence
                if word in top_words:
                    i = top_words.index(word)
                    vector += truncated_svd[i]
                    cnt_words += 1
            if cnt_words != 0:
                vector /= cnt_words
```

```
            avg_w2v_vectors_preprocessed_project_title_tr.append(vector)

        print(len(avg_w2v_vectors_preprocessed_project_title_tr))
        print(len(avg_w2v_vectors_preprocessed_project_title_tr[0]))
```

```
100%|| 16000/16000 [00:01<00:00, 13925.21it/s]

16000
220
```

```
In [0]: # average Word2Vec
        # compute average word2vec for each review.
        avg_w2v_vectors_preprocessed_project_title_te = []; # the avg-w2v for each sentence/re
        for sentence in tqdm(test_preprocessed_titles): # for each review/sentence
            vector = np.zeros(220) # as word vectors are of zero length
            cnt_words =0; # num of words with a valid vector in the sentence/review
            for word in sentence.split(): # for each word in a review/sentence
                if word in top_words:
                    i = top_words.index(word)
                    vector += truncated_svd[i]
                    cnt_words += 1
            if cnt_words != 0:
                vector /= cnt_words
            avg_w2v_vectors_preprocessed_project_title_te.append(vector)

        print(len(avg_w2v_vectors_preprocessed_project_title_te))
        print(len(avg_w2v_vectors_preprocessed_project_title_te[0]))
```

```
100%|| 4000/4000 [00:00<00:00, 14236.04it/s]

4000
220
```

```
In [0]: from scipy.sparse import hstack
        X_tr = hstack((avg_w2v_vectors_tr,avg_w2v_vectors_preprocessed_project_title_tr ,tr_pr
                       X_train_cl_subcategories_ohe, X_train_school_state_ohe, X_train_teacher_
        X_te = hstack((avg_w2v_vectors_te, avg_w2v_vectors_preprocessed_project_title_te, te_pr
                       X_test_cl_subcategories_ohe, X_test_school_state_ohe, X_test_teacher_pre
```

```
In [0]: print("Final Data matrix")
        print(X_tr.shape, y_train.shape)
        print(X_te.shape, y_test.shape)
        print("="*100)
```

```
Final Data matrix
(16000, 542) (16000,)
(4000, 542) (4000,)
```

==========

2.5 Apply XGBoost on the Final Features from the above section
https://xgboost.readthedocs.io/en/latest/python/python_intro.html

```
In [0]: from lightgbm import LGBMClassifier
        from sklearn.model_selection import RandomizedSearchCV
        param1 = {'n_estimators':  [50,100,200,500,1000] ,
                  'max_depth' :  [10,15,20,25] ,
                  'reg_lambda':  [0.05,0.5,0,1,2] ,
                  'reg_alpha' :  [0.05,0.5,0,1,2] ,
                  'learning_rate':  [0.005,0.05,0.5,0.1]}


        estimator1 = LGBMClassifier(objective = "binary" ,eval_metric= 'auc',class_weight = "ba
        clf1= RandomizedSearchCV(estimator1, param_distributions=param1, scoring='roc_auc', cv=
        clf1.fit(X_tr,y_train)
```

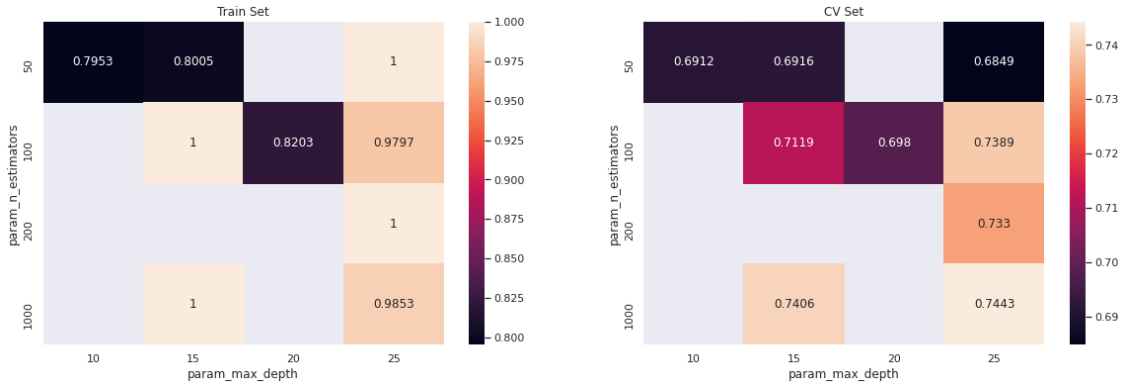
```
Out[0]: RandomizedSearchCV(cv=5, error_score=nan,
                           estimator=LGBMClassifier(boosting_type='gbdt',
                                                    class_weight='balanced',
                                                    colsample_bytree=1.0,
                                                    eval_metric='auc',
                                                    importance_type='split',
                                                    learning_rate=0.1, max_depth=-1,
                                                    min_child_samples=20,
                                                    min_child_weight=0.001,
                                                    min_split_gain=0.0,
                                                    n_estimators=100, n_jobs=-1,
                                                    num_leaves=31, objective='binary',
                                                    random_state=None, reg_a...
                                                    subsample_for_bin=200000,
                                                    subsample_freq=0),
                           iid='deprecated', n_iter=10, n_jobs=None,
                           param_distributions={'learning_rate': [0.005, 0.05, 0.5,
                                                                  0.1],
                                               'max_depth': [10, 15, 20, 25],
                                               'n_estimators': [50, 100, 200, 500,
                                                                1000],
                                               'reg_alpha': [0.05, 0.5, 0, 1, 2],
                                               'reg_lambda': [0.05, 0.5, 0, 1, 2]},
                           pre_dispatch='2*n_jobs', random_state=None, refit=True,
                           return_train_score=True, scoring='roc_auc', verbose=0)
```

```
In [0]: #https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchC
        a1=clf1.best_params_['n_estimators']
```

```
        p1 = clf1.best_params_['max_depth']
        q1 = clf1.best_params_['reg_lambda']
        r1 = clf1.best_params_['reg_alpha']
        s1 = clf1.best_params_['learning_rate']
        print(clf1.best_score_)
        print(a1)
        print(p1)
        print(q1)
        print(r1)
        print(s1)
```

```
0.7442511679086486
1000
25
0.5
1
0.005
```

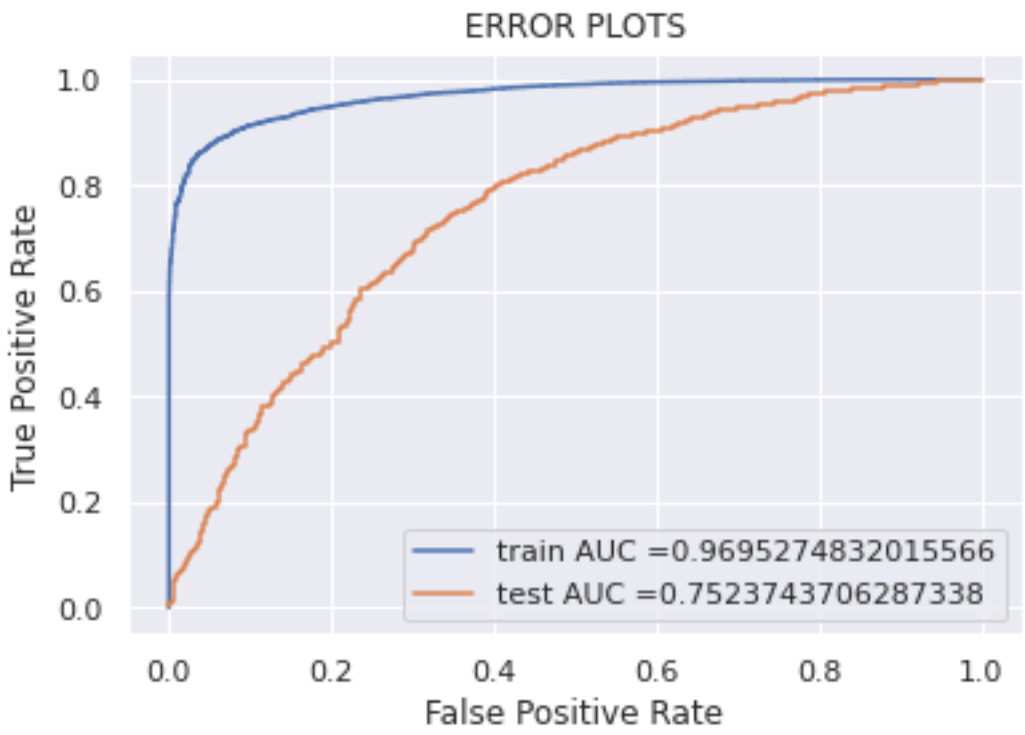
In [0]:
```
#https://towardsdatascience.com/using-3d-visualizations-to-tune-hyperparameters-of-ml-
#https://github.com/xoelop/Medium-posts/blob/master/3d%20cross%20validation/ML%206%20-
import seaborn as sns; sns.set()
max_scores1 = pd.DataFrame(clf1.cv_results_).groupby(['param_n_estimators', 'param_max_
fig, ax = plt.subplots(1,2, figsize=(20,6))
sns.heatmap(max_scores1.mean_train_score, annot = True, fmt='.4g', ax=ax[0])
sns.heatmap(max_scores1.mean_test_score, annot = True, fmt='.4g', ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



In [0]:
```
# Train new model
LG_svd = LGBMClassifier(max_depth=p1, n_estimators=a1, learning_rate = s1, reg_lambda =
LG_svd.fit(X_tr,y_train)
```

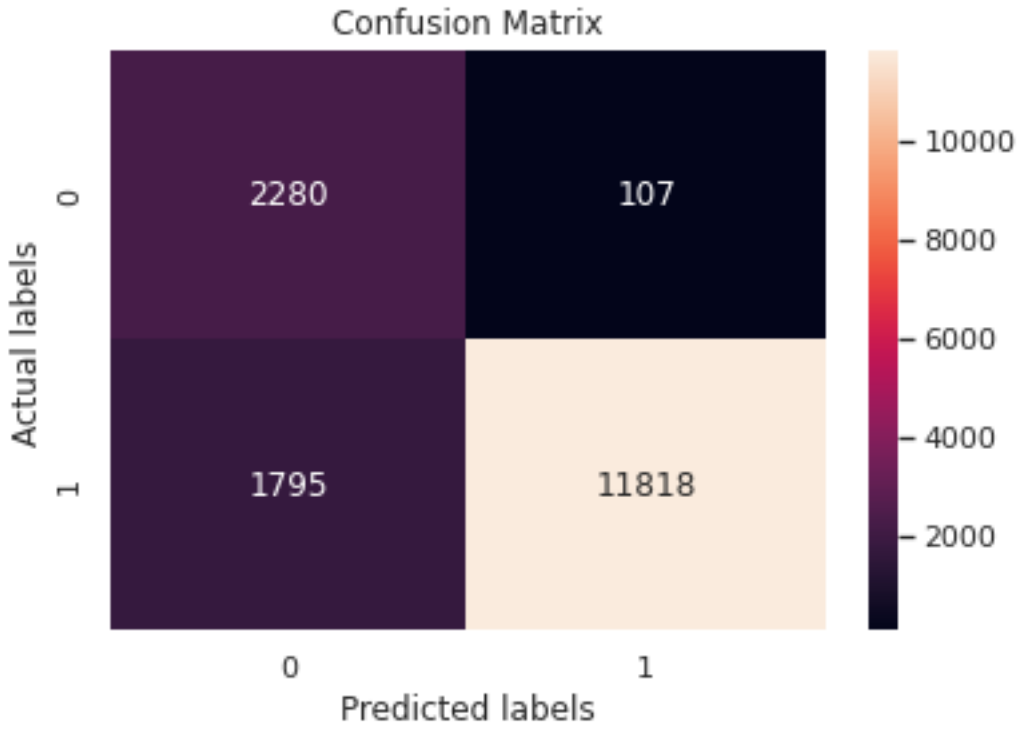
```
Out[0]: LGBMClassifier(boosting_type='gbdt', class_weight='balanced',
                       colsample_bytree=1.0, importance_type='split',
                       learning_rate=0.005, max_depth=25, min_child_samples=20,
                       min_child_weight=0.001, min_split_gain=0.0, n_estimators=1000,
                       n_jobs=-1, num_leaves=31, objective=None, random_state=None,
                       reg_alpha=1, reg_lambda=0.5, silent=True, subsample=1.0,
                       subsample_for_bin=200000, subsample_freq=0)
```

```
In [0]: #https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.h
        y_train_pred1 = LG_svd.predict_proba(X_tr) [:,1]
        y_test_pred1 = LG_svd.predict_proba(X_te) [:,1]

        train_fpr1, train_tpr1, tr_thresholds1 = roc_curve(y_train, y_train_pred1)
        test_fpr1, test_tpr1, te_thresholds1 = roc_curve(y_test, y_test_pred1)
        plt.plot(train_fpr1, train_tpr1, label="train AUC ="+str(auc(train_fpr1, train_tpr1)))
        plt.plot(test_fpr1, test_tpr1, label="test AUC ="+str(auc(test_fpr1, test_tpr1)))
        plt.legend()

        plt.xlabel("False Positive Rate")
        plt.ylabel("True Positive Rate")
        plt.title("ERROR PLOTS")
        plt.grid(True)
        plt.show()
```



In [0]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#Train dataset - COnfusion Matrix
import seaborn as sns
import matplotlib.pyplot as plt
print("Confusin Matrix On train")
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_train, LG_svd.predict(X_tr)), annot=True, ax = ax,fmt='g

# labels, title and ticks
ax.set_xlabel('Predicted labels');
ax.set_ylabel('Actual labels');
ax.set_title('Confusion Matrix');
```

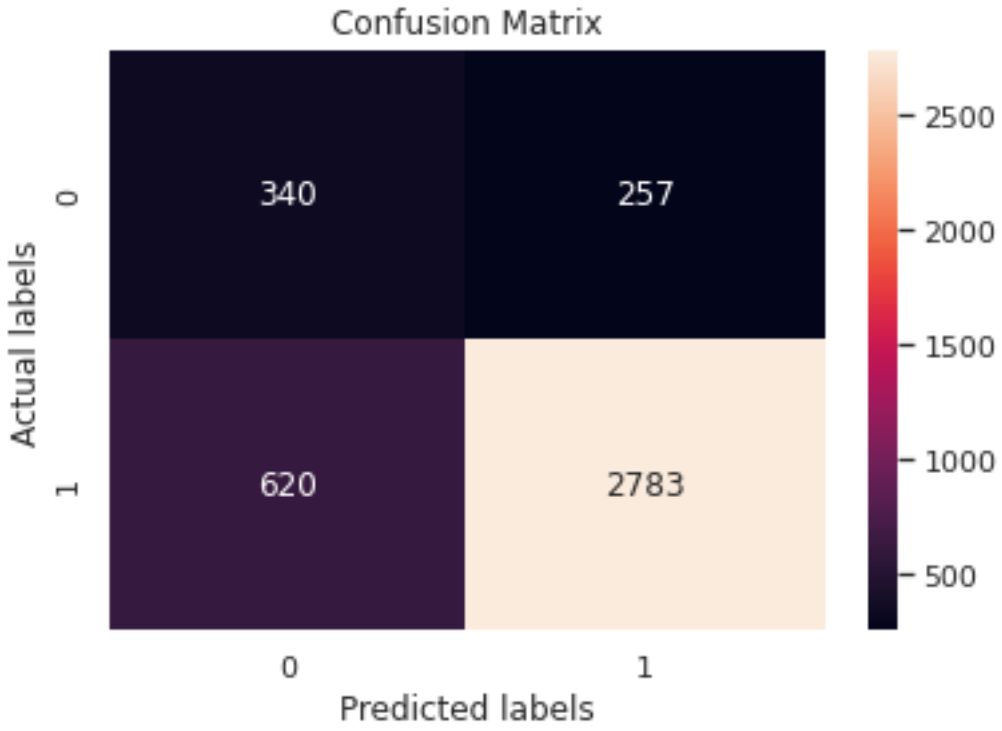
```
Confusin Matrix On train
```



In [0]:
```
#https://stackoverflow.com/questions/35572000/how-can-i-plot-a-confusion-matrix
#Train dataset - COnfusion Matrix
import seaborn as sns
import matplotlib.pyplot as plt
print("Confusin Matrix On test")
ax= plt.subplot()
sns.heatmap(confusion_matrix(y_test, LG_svd.predict(X_te)), annot=True, ax = ax,fmt='g
```

```
        # labels, title and ticks
        ax.set_xlabel('Predicted labels');
        ax.set_ylabel('Actual labels');
        ax.set_title('Confusion Matrix');
```

Confusin Matrix On test



3. Conclusion

```
In [0]: from prettytable import PrettyTable

        x = PrettyTable()

        x.field_names = ["vectorizer", "Model", "Max Depth ","n_estimators" ,"AUC"]

        x.add_row(["Covariance matrix", "Truncated SVD + XG Boost",25, 1000, 0.752])


        print(x)
```

```
+------------------+--------------------------+------------+--------------+-------+
|    vectorizer    |          Model           | Max Depth  | n_estimators |  AUC  |
```

Covariance matrix	Truncated SVD + XG Boost	25	1000	0.752

Toy example - to calculate co-occurence matrix

```
In [0]: #https://ideone.com/Ftofgp
        import numpy as np
        corpus = ["abc def ijk pqr", "pqr klm opq", "lmn pqr xyz abc def pqr abc"]
        top_words = ["abc","pqr","def"]
        window_size = 2

        CM = np.zeros((3,3),np.int32)
        for sentences in (corpus):
                words = sentences.split()
                for i, word in enumerate(words):
                        if word in top_words:
                                #print(i, word)
                                #for j in range(max(i-window_size,0), min(i+window_size+1,len(
                                #print(range(max(i-window_size,0), min(i+window_size,len(words
                                for j in range(max(i-window_size,0), min(i+window_size,len(wor
                                        #print(j)
                                        if words[j] in top_words:
                                                #print(j, words[j])
                                                if j!=i:
                                                        #print("*"*15)
                                                        #print(top_words.index(word))
                                                        #print(top_words.index(words[j]))
                                                        CM[top_words.index(word),top_words.inde
                                                        #CM[top_words.index(words[j]), top_wor
                                                        #print("*"*15)
                                                else:
                                                        pass
                #else:
                        #pass
        print(CM)
```

```
[[0 3 3]
 [3 0 2]
 [3 2 0]]
```