DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom pro
number of volunteers is needed to manually screen each submission before it's approved to be poste

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

- How to scale current manual processes and resources to screen 500,000 projects so that they c
as possible
- How to increase the consistency of project vetting across different volunteers to improve the ex
- 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 submit
the text of project descriptions as well as additional metadata about the project, teacher, and school.
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 f • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`

Follow link (ctrl + click)

Feature	Description
`project_subject_categories`	One or more (comma-separated) subject categories for the project fr • 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**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the proje • Literacy • Literature & Writing, Social Sciences
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • My students need hands on literacy materi
`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`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:**
`teacher_prefix`	Follow link (ctrl + click) 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 tea

* 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 pr resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`

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 tl following:

- __project_essay_1:__ "Describe your students: What makes your students special? Specific deta neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your stu lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_e

```
%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
```

Follow link (ctrl + click)

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

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

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

from tqdm import tqdm
import os
```

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

```
D:\installed\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Win
  warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")
```

1.1 Reading Data

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')
```

```
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']
```

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

Follow link (ctrl + click)

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

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

1.2 preprocessing of `project_subject_categories`

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

# 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", "Ca
        if 'The' in j.split(): # this will split each of the catogory based on space "Math &
            j=j.replace('The','') # if we have the words "The" we are going to replace it wit
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

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

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

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

1.3 preprocessing of `project_subject_subcategories`

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

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

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

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
```

Follow link (ctrl + click)

```
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

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

1.3 Text preprocessing

```
# 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)


project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

Follow link (ctrl + click)

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


# printing some random reviews
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third la
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love lear
==================================================
How do you remember your days of school? Was it in a sterile environment with plain wall
==================================================
My kindergarten students have varied disabilities ranging from speech and language delay
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher demonstrates
==================================================
```

```
# 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
```

Follow link (ctrl + click)

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

```
My kindergarten students have varied disabilities ranging from speech and language delay
==================================================
```

```
# \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 delay
```

```
#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 delay
```

```
# 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", "yo
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll"
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'h
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'unt
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'dur
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', '
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'bo
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'ver
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'does
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
            'won', "won't", 'wouldn', "wouldn't"]
```

```
# 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', ' ')  Follow link (ctrl + click)
    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%|██████████████████████████████████████████████████████████████████████████████| 109248/1
```

```
# after preprocesing
preprocessed_essays[20000]
```

```
'my kindergarten students varied disabilities ranging speech language delays cognitive d
```

1.4 Preprocessing of `project_title`

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

1.5 Preparing data for models

```
project_data.columns
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay'],
      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

Follow link (ctrl + click)

1.5.1 Vectorizing Categorical data

- https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/handling-categorica

```
# 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, binary
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)
```

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

```
# we use count vectorizer to convert the values into one
```

```
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False, bi
sub_categories_one_hot = vectorizer.fit_transform(project_data['clean_subcategories'].values)
print(vectorizer.get_feature_names())
print("Shape of matrix after one hot encodig ",sub_categories_one_hot.shape)
```

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

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

1.5.2 Vectorizing Text data

1.5.2.1 Bag of words

```
# We are considering only the words which appeared in at least 10 documents(rows or projects)
vectorizer = CountVectorizer(min_df=10)
text_bow = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_bow.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

```
# you can vectorize the title also
# before you vectorize the title make sure you preprocess it
```

1.5.2.2 TFIDF vectorizer

Follow link (ctrl + click)

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(min_df=10)
text_tfidf = vectorizer.fit_transform(preprocessed_essays)
print("Shape of matrix after one hot encodig ",text_tfidf.shape)
```

```
Shape of matrix after one hot encodig  (109248, 16623)
```

1.5.2.3 Using Pretrained Models: Avg W2V

```
'''
# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039
def loadGloveModel(gloveFile):
    print ("Loading Glove Model")
    f = open(gloveFile,'r', encoding="utf8")
    model = {}
    for line in tqdm(f):
        splitLine = line.split()
        word = splitLine[0]
```

```
        embedding = np.array([float(val) for val in splitLine[1:]])
        model[word] = embedding
    print ("Done.",len(model)," words loaded!")
    return model
model = loadGloveModel('glove.42B.300d.txt')

# ============================
Output:

Loading Glove Model
1917495it [06:32, 4879.69it/s]
Done. 1917495  words loaded!

# ============================

words = []
for i in preproced_texts:
    words.extend(i.split(' '))

for i in preproced_titles:
    words.extend(i.split(' '))
print("all the words in the coupus", len(words))
words = set(words)
print("the unique words in the coupus", len(words))

inter_words = set(model.keys()).intersection(words)
print("The number of words that are present in both glove vectors and our coupus", \
      len(inter_words),"(",np.round(len(inter_words)/len(words)*100,3),"%)")

words_courpus = {}
words_glove = set(model.keys())      Follow link (ctrl + click)
for i in words:
    if i in words_glove:
        words_courpus[i] = model[i]
print("word 2 vec length", len(words_courpus))


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-

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


'''
```

```
'\n# Reading glove vectors in python: https://stackoverflow.com/a/38230349/4084039\ndef


# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-
# make sure you have the glove_vectors file
```

```
with open('glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())


# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in glove_words:
            vector += model[word]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors.append(vector)

print(len(avg_w2v_vectors))
print(len(avg_w2v_vectors[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 109248/10
109248
300
```

1.5.2.3 Using Pretrained Models: TFIDF weighted W2V

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(preprocessed_essays)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())


# average Word2Vec
# compute average word2vec for each review.
tfidf_w2v_vectors = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(preprocessed_essays): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if (word in glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf value((sentence.
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_w2v_vectors.append(vector)
```

Follow link (ctrl + click)

```
print(len(tfidf_w2v_vectors))
print(len(tfidf_w2v_vectors[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/1
109248
300
```

```
# Similarly you can vectorize for title also
```

1.5.3 Vectorizing Numerical features

```
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')
```

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# standardization sklearn: https://scikit-learn.org/stable/modules/generated/sklearn.preproce
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.   ... 399.
# 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 standard
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}"

Follow link (ctrl + click)
# Now standardize the data with above maen and variance.
price_standardized = price_scalar.transform(project_data['price'].values.reshape(-1, 1))
```

```
price_standardized
```

```
array([[0.00098843, 0.00191166, 0.00330448, ..., 0.00153418, 0.00046704,
        0.00070265]])
```

1.5.4 Merging all the above features

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

```
print(categories_one_hot.shape)
print(sub_categories_one_hot.shape)
print(text_bow.shape)
print(price_standardized.shape)
```

```
(109248, 9)
(109248, 30)
(109248, 16623)
(109248, 1)
```

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
X = hstack((categories_one_hot, sub_categories_one_hot, text_bow, price_standardized))
X.shape
```

```
(109248, 16663)
```

```
# please write all the code with proper documentation, and proper titles for each subsection
# 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
```

__ Computing Sentiment Scores__

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

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

Follow link (ctrl + click)

```
sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest stude
for learning my students learn in many different ways using all of our senses and multiple in
of techniques to help all my students succeed students in my class come from a variety of dif
for wonderful sharing of experiences and cultures including native americans our school is a
learners which can be seen through collaborative student project based learning in and out of
in my class love to work with hands on materials and have many different opportunities to pra
mastered having the social skills to work cooperatively with friends is a crucial aspect of t
montana is the perfect place to learn about agriculture and nutrition my students love to rol
in the early childhood classroom i have had several kids ask me can we try cooking with real
and create common core cooking lessons where we learn important math and writing concepts whi
food for snack time my students will have a grounded appreciation for the work that went into
of where the ingredients came from as well as how it is healthy for their bodies this project
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade
and mix up healthy plants from our classroom garden in the spring we will also create our own
shared with families students will gain math and literature skills as well as a life long enj
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: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
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

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

Assignment 10: Clustering

- step 1: Choose any vectorizer (data matrix) that you have worked in any of the assignments, and
- step 2: Choose any of the feature selection/reduction algorithms ex: selectkbest features, pretra selection etc and reduce the number of features to 5k features
- step 3: Apply all three kmeans, Agglomerative clustering, DBSCAN
  - **K-Means Clustering:**
    - Find the best 'k' using the elbow-knee method (plot k vs inertia_)
  - **Agglomerative Clustering:**
    - Apply agglomerative algorithm and try a different number of clusters like 2,5 etc.
    - You can take less data points (as this is very computationally expensive one) to perform take a considerable amount of time to run.
  - **DBSCAN Clustering:**
    - Find the best 'eps' using the elbow-knee method.
    - You can take a smaller sample size for this as well.
- step 4: Summarize each cluster by manually observing few points from each cluster.
- step 5: You need to plot the word cloud with essay text for each cluster for each of algorithms n

Follow link (ctrl + click)

2. Clustering

2.1 Choose the best data matrix on which you got the best AUC

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

import sqlite3
import pandas as pd
import numpy as np
```

```
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.preprocessing import normalize

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_notebook as tqdm1
from tqdm import tqdm
import time
import os

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

from sklearn.model_selection import train_test_split
```

Follow link (ctrl + click)

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

```
project_data = pd.read_csv('train_data.csv')
resource_data = pd.read_csv('resources.csv')


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']
```

Text preprocessing(1)

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

# 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", "Ca
        if 'The' in j.split(): # this will split each of the catogory based on space "Math &
            j=j.replace('The','') # if we have the words "The" we are going to replace it wit
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
        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())
```

Follow link (ctrl + click)

```
project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(5)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	

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

Follow link (ctrl + click)

```
Counter({'Literacy_Language': 52239,
         'History_Civics': 5914,
         'Health_Sports': 14223,
         'Math_Science': 41421,
         'SpecialNeeds': 13642,
         'AppliedLearning': 12135,
         'Music_Arts': 10293,
         'Warmth': 1388,
         'Care_Hunger': 1388})
```

```
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))


# ind = np.arange(len(sorted_cat_dict))
```

```
# plt.figure(figsize=(20,5))
# p1 = plt.bar(ind, list(sorted_cat_dict.values()))

# plt.ylabel('Projects')
# plt.title('% of projects aproved category wise')
# plt.xticks(ind, list(sorted_cat_dict.keys()))
# plt.show()
# print(sorted_cat_dict)


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

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


project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Follow link (ctrl + click)

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

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

```
    my_counter.update(word.split())

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))


# ind = np.arange(len(sorted_sub_cat_dict))
# plt.figure(figsize=(20,5))
# p1 = plt.bar(ind, list(sorted_sub_cat_dict.values()))

# plt.ylabel('Projects')
# plt.title('% of projects aproved state wise')
# plt.xticks(ind, list(sorted_sub_cat_dict.keys()))
# plt.show()


# 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)


# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-grou
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

Follow link (ctrl + click)

```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')


#presence of the numerical digits in a strings with numeric : https://stackoverflow.com/a/198
def hasNumbers(inputString):
    return any(i.isdigit() for i in inputString)
p1 = project_data[['id','project_resource_summary']]
p1 = pd.DataFrame(data=p1)
p1.columns = ['id','digits_in_summary']
p1['digits_in_summary'] =  p1['digits_in_summary'].map(hasNumbers)
# https://stackoverflow.com/a/17383325/8089731
p1['digits_in_summary'] = p1['digits_in_summary'].astype(int)
project_data = pd.merge(project_data, p1, on='id', how='left')
project_data.head(5)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	
4	172407	p104768	be1f7507a[illegible]39ec	Mrs.	TX	

Follow link (ctrl + click)

5 rows × 21 columns

Text preprocessing(2)

```
# 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


# 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", "yo
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they',
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll"
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'h
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'unt
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'dur
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', '
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'bo
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'ver
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'does
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
            'won', "won't", 'wouldn', "wouldn't"]


# Combining all the above statemennts
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)
    sent = re.sub('nannan', '', 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())
```

Follow link (ctrl + click)


```
100%|██████████████████████████████████████████████████████████████████████████| 109248/10
```

```
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for title in tqdm(project_data['project_title'].values):
    _title = decontracted(title)
    title =  title.replace('\\r', ' ')
```

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

```
100%|██████████████████████████████████████████████████████████████████████████| 109248/109
```

```
preprocessed_titles[1000]
```

```
'sailing into super 4th grade year'
```

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

# 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

project_grade_cat_list = []
for i in tqdm1(project_grade_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", "Ca
        if 'The' in j.split(): # this will split each of the catogory based on space "Math &
            j=j.replace('The','') # if we have the words "The" we are going to replace it wit
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math &
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') Follow link (ctrl + click)
    project_grade_cat_list.append(temp.strip())
```

```
HBox(children=(IntProgress(value=0, max=109248), HTML(value='')))
```

```
project_data['clean_project_grade_category'] = project_grade_cat_list
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	

2 rows × 21 columns

```
project_data.drop(['project_essay_1','project_essay_2','project_essay_3','project_essay_4'],
project_data.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	pr
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	


```
#Replacing Nan's with maximum occured value: https://stackoverflow.com/a/51053916/8089731
project_data['teacher_prefix'].value_counts().argmax()
project_data.fillna(value=project_data['teacher_prefix'].value_counts().argmax(),axis=1,inpla

project_data['preprocessed_essays'] = preprocessed_essays
project_data['preprocessed_titles'] = preprocessed_titles
```

```
project_data.columns
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
       'digits_in_summary', 'clean_project_grade_category',
       'preprocessed_essays', 'preprocessed_titles'],
      dtype='object')
```

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

2.2 Make Data Model Ready: encoding numerical, categorical features

```
X_train, X_test, y_train, y_test = train_test_split(project_data,project_data['project_is_app
# X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=

X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
# X_cv.drop(['project_is_approved'], axis=1, inplace=True)
print(X_train.shape)
print(X_test.shape)
```

Follow link (ctrl + click)

```
(73196, 18)
(36052, 18)
```

1.4.1 Vectorizing Categorical data

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_cat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, bi
vectorizer_cat.fit(X_train['clean_categories'].values)
print(vectorizer_cat.get_feature_names())


categories_one_hot_train = vectorizer_cat.transform(X_train['clean_categories'].values)
# categories_one_hot_cv = vectorizer_cat.transform(X_cv['clean_categories'].values)
categories_one_hot_test = vectorizer_cat.transform(X_test['clean_categories'].values)
print("Shape of matrix after one hot encodig_train ",categories_one_hot_train.shape)
```

```
# print("Shape of matrix after one hot encodig_cv ",categories_one_hot_cv.shape)
print("Shape of matrix after one hot encodig_test ",categories_one_hot_test.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNee
Shape of matrix after one hot encodig_train  (73196, 9)
Shape of matrix after one hot encodig_test  (36052, 9)
```

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer_sub_cat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=F
vectorizer_sub_cat.fit(X_train['clean_subcategories'].values)
print(vectorizer_sub_cat.get_feature_names())


sub_categories_one_hot_train = vectorizer_sub_cat.transform(X_train['clean_subcategories'].va
# sub_categories_one_hot_cv = vectorizer_sub_cat.transform(X_cv['clean_subcategories'].values
sub_categories_one_hot_test = vectorizer_sub_cat.transform(X_test['clean_subcategories'].valu
print("Shape of matrix after one hot encodig_train ",sub_categories_one_hot_train.shape)
# print("Shape of matrix after one hot encodig_cv ",sub_categories_one_hot_cv.shape)
print("Shape of matrix after one hot encodig_test ",sub_categories_one_hot_test.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurric
Shape of matrix after one hot encodig_train  (73196, 30)
Shape of matrix after one hot encodig_test  (36052, 30)
```

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_state = CountVectorizer( lowercase=False, binary=True)
vectorizer_state.fit(X_train['school_state'].values)
print(vectorizer_state.get_feature_names())
```

Follow link (ctrl + click)

```
school_state_one_hot_train = vectorizer_state.transform(X_train['school_state'].values)
# school_state_one_hot_cv = vectorizer_state.transform(X_cv['school_state'].values)
school_state_one_hot_test = vectorizer_state.transform(X_test['school_state'].values)
print("Shape of matrix after one hot encodig_train ",school_state_one_hot_train.shape)
# print("Shape of matrix after one hot encodig_cv ",school_state_one_hot_cv.shape)
print("Shape of matrix after one hot encodig_test ",school_state_one_hot_test.shape)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL
Shape of matrix after one hot encodig_train  (73196, 51)
Shape of matrix after one hot encodig_test  (36052, 51)
```

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_teacherprefix = CountVectorizer( lowercase=False, binary=True)
vectorizer_teacherprefix.fit(X_train['teacher_prefix'].values.astype('U'))
print(vectorizer_teacherprefix.get_feature_names())

#https://stackoverflow.com/a/39308809/8089731
```

```
teacher_prefix_one_hot_train = vectorizer_teacherprefix.transform(X_train['teacher_prefix'].v
# teacher_prefix_one_hot_cv = vectorizer_teacherprefix.transform(X_cv['teacher_prefix'].value
teacher_prefix_one_hot_test = vectorizer_teacherprefix.transform(X_test['teacher_prefix'].val
print("Shape of matrix after one hot encodig_train ",teacher_prefix_one_hot_train.shape)
# print("Shape of matrix after one hot encodig_cv ",teacher_prefix_one_hot_cv.shape)
print("Shape of matrix after one hot encodig_test ",teacher_prefix_one_hot_test[:5,:])
# print(X_train['teacher_prefix'].value_counts())
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encodig_train  (73196, 5)
Shape of matrix after one hot encodig_test    (0, 3)    1
  (1, 2)        1
  (2, 3)        1
  (3, 4)        1
  (4, 1)        1
```

```
print(project_data['clean_project_grade_category'].unique())# we use count vectorizer to conv
from sklearn.feature_extraction.text import CountVectorizer
# https://stackoverflow.com/a/38161028/8089731
pattern = "(?u)\\b[\\w-]+\\b"
vectorizer_projectgrade = CountVectorizer(token_pattern=pattern, lowercase=False, binary=True
vectorizer_projectgrade.fit(X_train['clean_project_grade_category'].values)
print(vectorizer_projectgrade.get_feature_names())

#https://stackoverflow.com/a/39308809/8089731
project_grade_category_one_hot_train = vectorizer_projectgrade.transform(X_train['clean_proje
# project_grade_category_one_hot_cv = vectorizer_projectgrade.transform(X_cv['clean_project_g
project_grade_category_one_hot_test = vectorizer_projectgrade.transform(X_test['clean_project
print("Shape of matrix after one hot encodig_train ",project_grade_category_one_hot_train.sha
# print("Shape of matrix after one hot encodig_cv ",project_grade_category_one_hot_cv.shape)
print("Shape of matrix after one hot encodig_test ",project_grade_category_one_hot_test[:5,:]
```

Follow link (ctrl + click)

```
['GradesPreK-2' 'Grades6-8' 'Grades3-5' 'Grades9-12']
['Grades3-5', 'Grades6-8', 'Grades9-12', 'GradesPreK-2']
Shape of matrix after one hot encodig_train  (73196, 4)
Shape of matrix after one hot encodig_test    (0, 3)    1
  (1, 0)        1
  (2, 3)        1
  (3, 1)        1
  (4, 2)        1
```

Vectorizing Numerical features

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

# 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.   ... 399.
```

```
# Reshape your data either using array.reshape(-1, 1)

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

# train_text_feature_onehotCoding = normalize(train_text_feature_onehotCoding, axis=0)
# Now standardize the data with above maen and variance.
price_standardized_train = normalize(X_train['price'].values.reshape(-1, 1),axis=0)
# price_standardized_cv = price_scalar.transform(X_cv['price'].values.reshape(-1, 1))
price_standardized_test = normalize(X_test['price'].values.reshape(-1, 1),axis=0)
print(price_standardized_train.shape)
# print(price_standardized_cv.shape)
print(price_standardized_test.shape)
```

```
(73196, 1)
(36052, 1)
```

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

# 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.   ... 399.
# Reshape your data either using array.reshape(-1, 1)

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

Follow link (ctrl + click)

```
# Now standardize the data with above maen and variance.
quantity_standardized_train = normalize(X_train['quantity'].values.reshape(-1, 1),axis=0)
# quantity_standardized_cv = quantity_scalar.transform(X_cv['quantity'].values.reshape(-1, 1)
quantity_standardized_test = normalize(X_test['quantity'].values.reshape(-1, 1),axis=0)
print(quantity_standardized_train.shape)
# print(quantity_standardized_cv.shape)
print(quantity_standardized_test.shape)
```

```
(73196, 1)
(36052, 1)
```

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

# 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.   ... 399.
```

```
# 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(X_train['teacher_number_of_previous
# print(f"Mean : {teacher_number_of_previously_posted_projects_scalar.mean_[0]}, Standard dev

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized_train = normalize(X_train['teacher_
# teacher_number_of_previously_posted_projects_standardized_cv = teacher_number_of_previously
teacher_number_of_previously_posted_projects_standardized_test = normalize(X_test['teacher_nu
print(teacher_number_of_previously_posted_projects_standardized_train.shape)
# print(teacher_number_of_previously_posted_projects_standardized_cv.shape)
print(teacher_number_of_previously_posted_projects_standardized_test.shape)
```

```
(73196, 1)
(36052, 1)
```

```
# 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
```

2.3 Make Data Model Ready: encoding eassay, and project_title

```
X_train.head(2)
```

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state
4279	136257	p120950	18b42a21b08dda1e0788525af1163ab4	Ms.	MN
17572	100630	p037368	678f3b6ae314bb90fa888de642eed8b6	Mrs.	UT

TFIDF Vectorizer on `project_TEXT/ESSAYS` (Train,Cv,Test)

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_essays = TfidfVectorizer(min_df=10,max_features=5000,ngram_range=(1,2))
vectorizer_tfidf_essays.fit(X_train['preprocessed_essays'])

text_tfidf_train = vectorizer_tfidf_essays.transform(X_train['preprocessed_essays'])
# text_tfidf_cv = vectorizer_tfidf_essays.transform(X_cv['preprocessed_essays'])
text_tfidf_test = vectorizer_tfidf_essays.transform(X_test['preprocessed_essays'])
print("Shape of matrix after tfidf_text_train ",text_tfidf_train.shape)
# print("Shape of matrix after tfidf_text_cv ",text_tfidf_cv.shape)
print("Shape of matrix after tfidf_text_test ",text_tfidf_test.shape)
```

```
Shape of matrix after tfidf_text_train  (73196, 5000)
Shape of matrix after tfidf_text_test  (36052, 5000)
```

TFIDF Vectorizer on `project_title` (Train,Cv,Test)

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer_tfidf_title = TfidfVectorizer(min_df=10)
vectorizer_tfidf_title.fit(X_train['preprocessed_titles'])

title_tfidf_train = vectorizer_tfidf_title.transform(X_train['preprocessed_titles'])
# title_tfidf_cv = vectorizer_tfidf_title.transform(X_cv['preprocessed_titles'])
title_tfidf_test = vectorizer_tfidf_title.transform(X_test['preprocessed_titles'])
print("Shape of matrix after tfidf_title_train ",title_tfidf_train.shape)
# print("Shape of matrix after tfidf_title_cv ",title_tfidf_cv.shape)
print("Shape of matrix after tfidf_title_test ",title_tfidf_test.shape)
```

```
Shape of matrix after tfidf_title_train  (73196, 2643)
Shape of matrix after tfidf_title_test  (36052, 2643)
```

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

import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

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

```
project_data.columns
```

```
      Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
             'project_submitted_datetime', 'project_title',
             'project_resource_summary',
             'teacher_number_of_previously_posted_projects', 'project_is_approved',
             'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
             'digits_in_summary', 'clean_project_grade_category',
             'preprocessed_essays', 'preprocessed_titles'],
            dtype='object')
```

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((categories_one_hot_train,sub_categories_one_hot_train,school_state_one_hot_tra
               ,project_grade_category_one_hot_train,price_standardized_train,quantity_standa
               ,teacher_number_of_previously_posted_projects_standardized_train,text_tfidf_tr
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_one_hot_cv,teac
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quantity_standardiz
#                ,teacher_number_of_previously_posted_projects_standardized_cv,text_tfidf_cv,
X_te = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_state_one_hot_test,
               ,project_grade_category_one_hot_test,price_standardized_test,quantity_standard
               ,teacher_number_of_previously_posted_projects_standardized_test,text_tfidf_tes

print("Final Data matrix on TFIDF")
print(X_tr.shape, y_train.shape)
# print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix on TFIDF
(73196, 7745) (73196,)
(36052, 7745) (36052,)
====================================================================================================
```

```
X_te.shape
```

```
(36052, 7745)
```

```
# 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
```

2.4 Dimensionality Reduction on the selected features

```
###########################################################################
# from sklearn.preprocessing import MaxAbsScaler
# scaler = MaxAbsScaler()
# X_tr = scaler.fit_transform(X_tr,y_train)
# X_te = scaler.transform(X_te)
###########################################################################
from sklearn.feature_selection import SelectKBest, chi2
t = SelectKBest(chi2,k=5000).fit(X_tr, y_train)
X_tr = t.transform(X_tr)
X_te = t.transform(X_te)
###########################################################################
print("Final Data matrix on TFIDF")
print(X_tr.shape, y_train.shape)
# print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix on TFIDF
(73196, 5000) (73196,)
(36052, 5000) (36052,)
====================================================================================================
```

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

2.5 Apply Kmeans

```
from sklearn.cluster import KMeans

k_values = [2,3,4,5,6,7,8]
loss = []
for i in k_values:
    kmeans = KMeans(n_clusters=i, n_jobs=-1).fit(X_tr)
    loss.append(kmeans.inertia_)


plt.plot(k_values, loss)
plt.xlabel('K',size=14)
```

```
plt.ylabel('Loss',size=14)
plt.title('Loss VS K Plot',size=18)
plt.grid()
plt.show()
```


```
optimal_k = 6

kmeans = KMeans(n_clusters=optimal_k, n_jobs=-1).fit(X_tr)
```

```
essays = X_train['preprocessed_essays'].values

cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []
cluster6 = []
for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(essays[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(essays[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(essays[i])
    elif  kmeans.labels_[i] == 3:
        cluster4.append(essays[i])
    elif  kmeans.labels_[i] == 4:
        cluster5.append(essays[i])
    elif  kmeans.labels_[i] == 5:
        cluster6.append(essays[i])
```

```
for i in range(3):
    print('%s\n'%(cluster1[i]))
```

within class i great diversity learners i never school including preschool kindergarten

we small rural school east texas high poverty rate low ses there no one way approach lea

belle hall awesome school full wonderful learners over 600 students come computer lab we

```
for i in range(3):
    print('%s\n'%(cluster2[i]))
```

my classroom slc consist low function students see world differently us they face challe

my students special many reasons there genuine thirst learning among students despite no

in class multiply disabled students camden new jersey daily learning overcome obstacles

```
for i in range(3):
    print('%s\n'%(cluster3[i]))
```

students come school eager learn explore they excited talk friends teacher at young age

as educator goal provide students positive learning experiences safe comfortable brain f

my students hispanic mainly central america asian mainly vietnam african american white

```
for i in range(3):
    print('%s\n'%(cluster4[i]))
```

our upper elementary school serves 3rd 5th graders south carolina our students come high

i one two pe teachers title 1 school located florence sc our students come us various ba

our students come across city philadelphia entire school qualifies free lunch transporta

```
for i in range(3):
    print('%s\n'%(cluster5[i]))
```

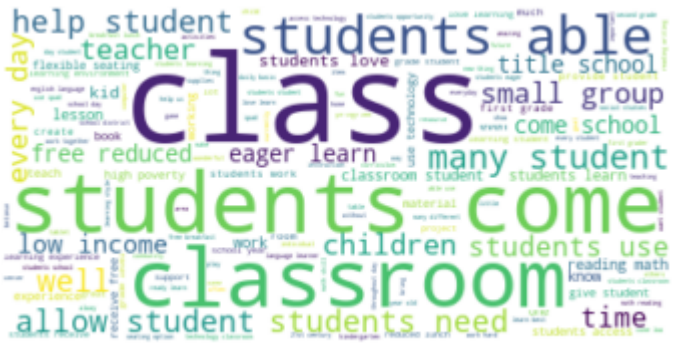
```
my students diverse socioeconomic backgrounds ethnicities they hardworking want best i w
as educator every experience classes smart literally extra homework keep our kindergarte
our school located rural community every student needs something different walk classroo
```

```
for i in range(3):
    print('%s\n'%(cluster6[i]))
```

```
welcome intellectually gifted classroom i teach highly gifted disabled gifted we class c
i work low income elementary school limited funding comes technology my students talente
our school inclusive high school located city one fastest growing areas northern califor
```

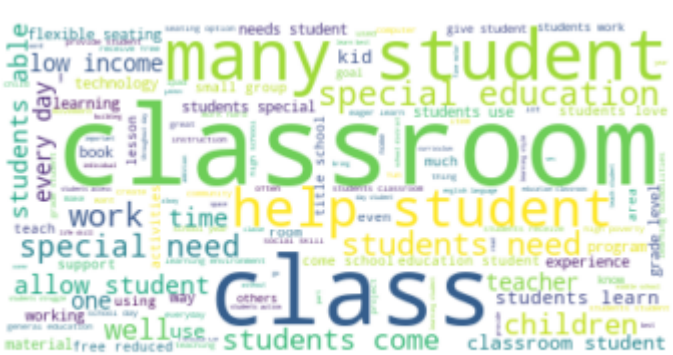
```
# for i in cluster1:
#     print(i.split())
# cluster1.values()
```

```
#cluster 1
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

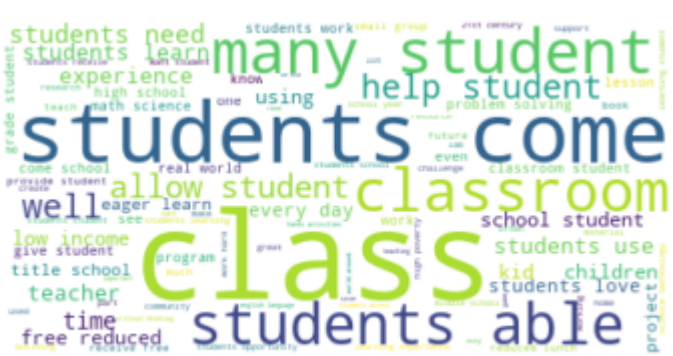



```
#cluster 2
words=''
for i in cluster2:
```

```
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

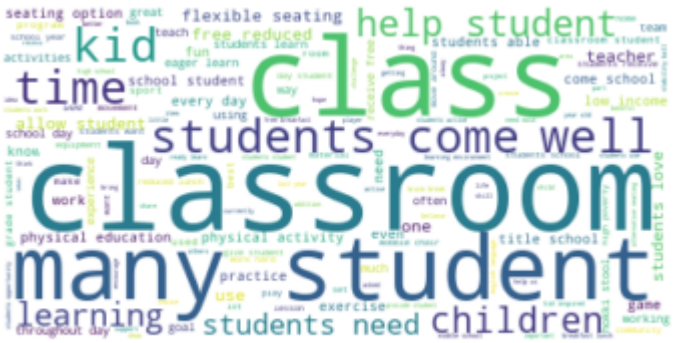



```
#cluster 3
words=''
for i in cluster3:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

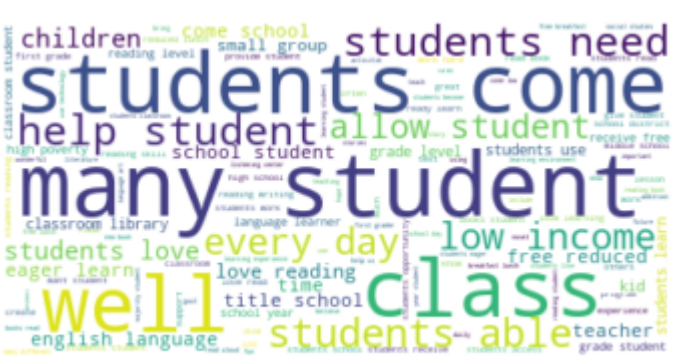



```
#cluster 4
words=''
for i in cluster4:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)
```

```
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

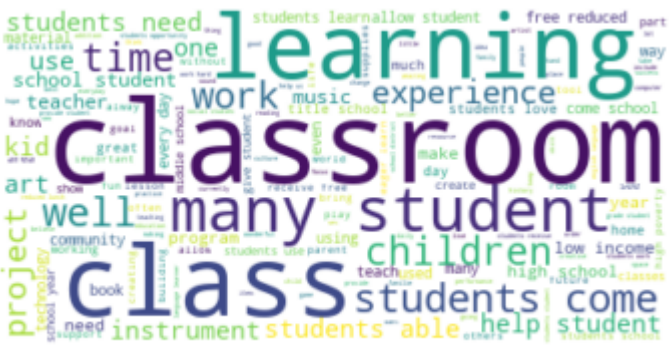



```
#cluster 5
words=''
for i in cluster5:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```




```
#cluster 6
words=''
for i in cluster6:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
```

```
plt.show()
```




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

2.6 Apply AgglomerativeClustering

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

import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

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

```
project_data.columns
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
       'digits_in_summary', 'clean_project_grade_category',
       'preprocessed_essays', 'preprocessed_titles'],
      dtype='object')
```

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((categories_one_hot_train,sub_categories_one_hot_train,school_state_one_hot_tra
               ,project_grade_category_one_hot_train,price_standardized_train,quantity_standa
               ,teacher_number_of_previously_posted_projects_standardized_train,text_tfidf_tr
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_one_hot_cv,teac
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quantity_standardiz
#                ,teacher_number_of_previously_posted_projects_standardized_cv,text_tfidf_cv,
X_te = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_state_one_hot_test,
               ,project_grade_category_one_hot_test,price_standardized_test,quantity_standard
               ,teacher_number_of_previously_posted_projects_standardized_test,text_tfidf_tes

print("Final Data matrix on TFIDF")
print(X_tr.shape, y_train.shape)
# print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix on TFIDF
(73196, 7745) (73196,)
(36052, 7745) (36052,)
====================================================================================================
```

```
X_te.shape
```

```
(36052, 7745)
```

```
# 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
```

2.4 Dimensionality Reduction on the selected features

```
##########################################################################
# from sklearn.preprocessing import MaxAbsScaler
# scaler = MaxAbsScaler()
# X_tr = scaler.fit_transform(X_tr,y_train)
# X_te = scaler.transform(X_te)
##########################################################################
from sklearn.feature_selection import SelectKBest, chi2
```

```
t = SelectKBest(chi2,k=5000).fit(X_tr, y_train)
X_tr = t.transform(X_tr)
X_te = t.transform(X_te)
#########################################################################
print("Final Data matrix on TFIDF")
print(X_tr.shape, y_train.shape)
# print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix on TFIDF
(73196, 5000) (73196,)
(36052, 5000) (36052,)
====================================================================================================
```

```
X_tr = X_tr[:5000]
X_train = X_train[:5000]
```

```
X_tr.shape
```

```
(5000, 5000)
```

for k=2

```
from sklearn.cluster import AgglomerativeClustering

aggcl=AgglomerativeClustering(n_clusters=2).fit(X_tr.toarray())
```

```
cluster1=[]
cluster2=[]
essays = X_train['preprocessed_essays'].values
for i in range(aggcl.labels_.shape[0]):
    if aggcl.labels_[i] == 0:
        cluster1.append(essays[i])
    elif aggcl.labels_[i] == 1:
        cluster2.append(essays[i])
```

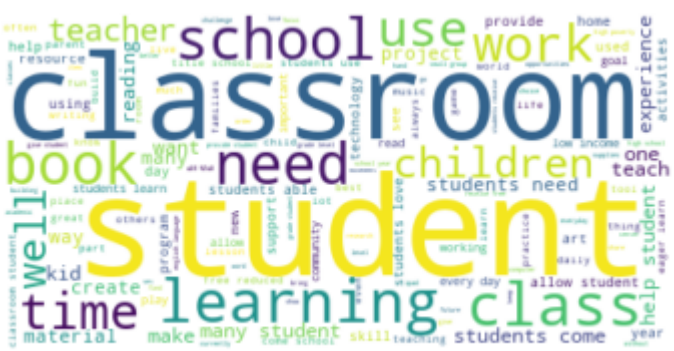
```
for i in range(3):
    print('%s\n'%(cluster1[i]))
```

```
 my students diverse socioeconomic backgrounds ethnicities they hardworking want best i w
```

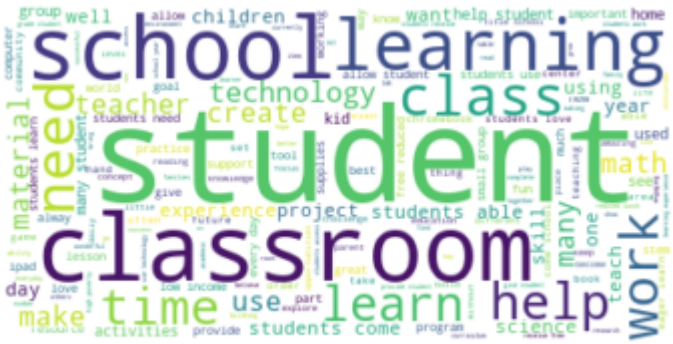
```
for i in range(3):
    print('%s\n'%(cluster2[i]))
```

```
 within class i great diversity learners i never school including preschool kindergarten

 we small rural school east texas high poverty rate low ses there no one way approach lea

 belle hall awesome school full wonderful learners over 600 students come computer lab we
```

```
#cluster 1
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```




```
#cluster 2
words=''
for i in cluster2:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



for k=5

```
from sklearn.cluster import AgglomerativeClustering

aggcl=AgglomerativeClustering(n_clusters=5).fit(X_tr.toarray())


cluster1=[]
cluster2=[]
cluster3=[]
cluster4=[]
cluster5=[]
essays = X_train['preprocessed_essays'].values
for i in range(aggcl.labels_.shape[0]):
    if aggcl.labels_[i] == 0:
        cluster1.append(essays[i])
    elif aggcl.labels_[i] == 1:
        cluster2.append(essays[i])
    elif aggcl.labels_[i] == 2:
        cluster3.append(essays[i])
    elif  aggcl.labels_[i] == 3:
        cluster4.append(essays[i])
    elif  aggcl.labels_[i] == 4:
        cluster5.append(essays[i])




for i in range(3):
    print('%s\n'%(cluster1[i]))
```

```
within class i great diversity learners i seven school including preschool kindergarten
```

```
for i in range(3):
    print('%s\n'%(cluster2[i]))
```

```
welcome intellectually gifted classroom i teach highly gifted disabled gifted we class c
i work low income elementary school limited funding comes technology my students talente
students come school eager learn explore they excited talk friends teacher at young age
```

```
for i in range(3):
    print('%s\n'%(cluster3[i]))
```

```
my students diverse socioeconomic backgrounds ethnicities they hardworking want best i w
as educator every experience classes smart literally extra homework keep our kindergarte
our school located rural community every student needs something different walk classroo
```

```
for i in range(3):
    print('%s\n'%(cluster4[i]))
```

```
my classroom slc consist low function students see world differently us they face challe
in class multiply disabled students camden new jersey daily learning overcome obstacles
my students come school day eager learn the school title i school means 99 students pove
```

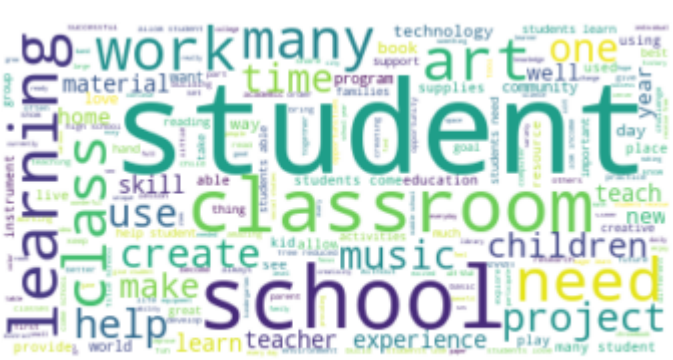
```
for i in range(3):
    print('%s\n'%(cluster5[i]))
```

```
our upper elementary school serves 3rd 5th graders south carolina our students come high
i one two pe teachers title 1 school located florence sc our students come us various ba
our students come across city philadelphia entire school qualifies free lunch transporta
```

```
#cluster 1
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)
```

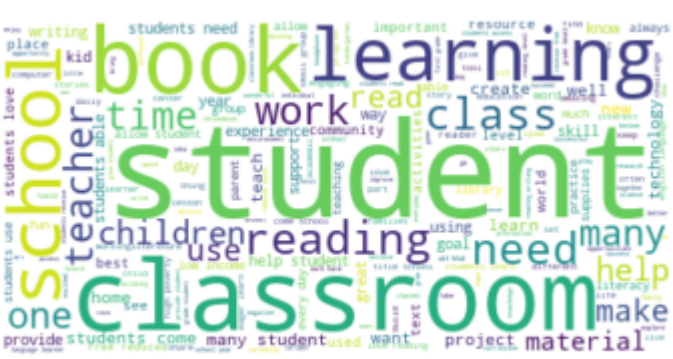
```
# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```


```
#cluster 2
words=''
for i in cluster2:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

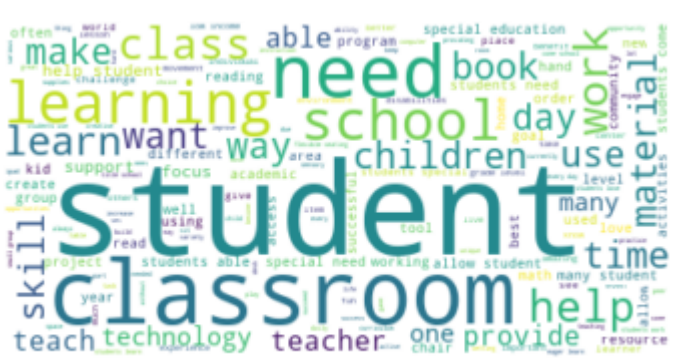



```
#cluster 3
words=''
for i in cluster3:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
```

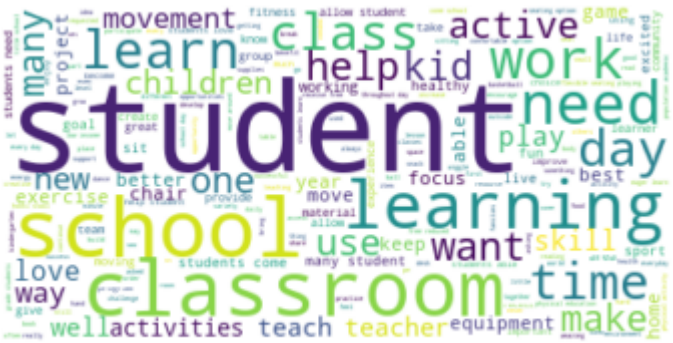
```
plt.show()
```




```
#cluster 4
words=''
for i in cluster4:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```




```
#cluster 5
words=''
for i in cluster5:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```




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

2.7 Apply DBSCAN

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

import dill
# dill.dump_session('notebook_env.db')
dill.load_session('notebook_env.db')
```

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

```
project_data.columns
```

```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_title',
       'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price', 'quantity',
       'digits_in_summary', 'clean_project_grade_category',
       'preprocessed_essays', 'preprocessed_titles'],
      dtype='object')
```

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr = hstack((categories_one_hot_train,sub_categories_one_hot_train,school_state_one_hot_tra
               ,project_grade_category_one_hot_train,price_standardized_train,quantity_standa
               ,teacher_number_of_previously_posted_projects_standardized_train,text_tfidf_tr
# X_cr = hstack((categories_one_hot_cv,sub_categories_one_hot_cv,school_state_one_hot_cv,teac
#                ,project_grade_category_one_hot_cv,price_standardized_cv,quantity_standardiz
#                ,teacher_number_of_previously_posted_projects_standardized_cv,text_tfidf_cv,
X_te = hstack((categories_one_hot_test,sub_categories_one_hot_test,school_state_one_hot_test,
               ,project_grade_category_one_hot_test,price_standardized_test,quantity_standard
               ,teacher_number_of_previously_posted_projects_standardized_test,text_tfidf_tes

print("Final Data matrix on TFIDF")
print(X_tr.shape, y_train.shape)
# print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix on TFIDF
(73196, 7745) (73196,)
(36052, 7745) (36052,)
====================================================================================================
```

```
X_te.shape
```

```
(36052, 7745)
```

```
# 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
```

2.4 Dimensionality Reduction on the selected features

```
########################################################################
# from sklearn.preprocessing import MaxAbsScaler
# scaler = MaxAbsScaler()
# X_tr = scaler.fit_transform(X_tr,y_train)
# X_te = scaler.transform(X_te)
```

```
#########################################################################
from sklearn.feature_selection import SelectKBest, chi2
t = SelectKBest(chi2,k=5000).fit(X_tr, y_train)
X_tr = t.transform(X_tr)
X_te = t.transform(X_te)
#########################################################################
print("Final Data matrix on TFIDF")
print(X_tr.shape, y_train.shape)
# print(X_cr.shape, y_cv.shape)
print(X_te.shape, y_test.shape)
print("="*100)
```

```
Final Data matrix on TFIDF
(73196, 5000) (73196,)
(36052, 5000) (36052,)
====================================================================================================
```

```
X_tr = X_tr[:5000]
X_train = X_train[:5000]
```

```
X_tr.shape
```

```
(5000, 5000)
```

```
from sklearn.preprocessing import StandardScaler
# dat=StandardScaler().fit_transform(X_tr.toarray())
dat=X_tr.toarray()
dat
```

```
array([[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.]])
```

```
from sklearn.metrics.pairwise import euclidean_distances

euclidean_distances(dat, dat[1].reshape(1, -1))
```

```
array([[3.13236693e+00],
       [4.21468485e-08],
       [3.20124722e+00],
       ...,
       [3.10522436e+00],
       [3.27882999e+00],
       [3.19873778e+00]])
```

```
# sorted_distance = np.sort(np.array(distance))
# len(sorted_distance)
```

```
806
```

```
# temp=np.sort(np.sum((dat-point),axis=1),axis=None)
# (temp).shape
```

```
(5000,)
```

```
# np.sum((dat-point),axis=1)
```

```
array([1.64712652, 2.30341677, 1.58066117, ..., 2.93591794, 1.03866231,
       0.        ])
```

```
# tt=(dat-point)**2
# tt[2000]
```

```
array([0., 0., 0., ..., 0., 0., 0.])
```

```
# np.sum((dat-point)**2,axis=1)
```

```
array([11.70919891, 10.23192341, 10.30200722, ..., 11.59827737,
        8.73397533,  0.        ])
```

```
# distance
```

```
      [array([104.26679466]),
       array([97.61290802]),
       array([92.24469259]),
       array([94.72413144]),
       array([91.92699148]),
       array([101.63258988]),
       array([94.72665935]),
       array([111.4489379]),
       array([92.75662884]),
       array([94.62546444]),
       array([106.28762275]),
       array([113.50326139]),
       array([104.62735836]),
       array([95.92549774]),
       array([98.26058222]),
       array([109.9701546]),
       array([91.7236505]),
       array([104.21250268]),
       array([90.0890306]),
       array([117.76242536]),
       array([94.69313352]),
       array([91.97822737]),
       array([92.95772527]),
       array([95.75440967]),
       array([91.00358232]),
       array([125.25245375]),
       array([96.55960104]),
       array([93.61834726]),
       array([99.3178604]),
       array([110.71166597]),
       array([90.82297469]),
       array([95.86445436]),
       array([84.95199019]),
       array([93.56511226]),
       array([99.80492315]),
       array([95.50944315]),
       array([99.72099326]),
       array([96.18770598]),
       array([97.21200634])]
```

```
from sklearn.metrics.pairwise import euclidean_distances

euclidean_distances(dat, dat[464].reshape(1, -1))
```

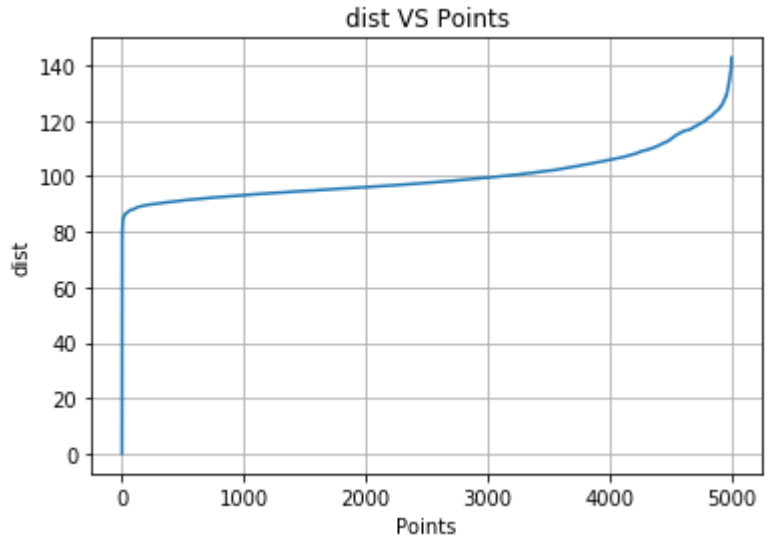
```
array([[3.6456085 ],
       [2.77862866],
       [3.41598141],
       ...,
       [2.60627774],
       [3.15575795],
       [3.12398718]])
```

```
# np.sort(sorted_dist[:50].reshape(1,-1))[0]
```

```
array([ 84.95199019,  90.0890306 ,  90.82297469,  91.00358232,
        91.7236505 ,  91.92699148,  91.97822737,  92.24469259,
        92.75662884,  92.91832753,  92.95772527,  93.56511226,
        93.61834726,  94.5552292 ,  94.62546444,  94.69313352,
        94.72413144,  94.72665935,  95.48144355,  95.50944315,
        95.75440967,  95.86445436,  95.92549774,  96.18770598,
        96.33094241,  96.41486617,  96.55960104,  97.21200634,
        97.61290802,  98.26058222,  99.3178604 ,  99.66575521,
        99.72099326,  99.80492315, 101.63258988, 103.80911931,
       104.21250268, 104.26679466, 104.62735836, 106.28762275,
       108.84403901, 109.9701546 , 110.71166597, 111.4489379 ,
       112.27431254, 113.50326139, 116.28452388, 116.53726028,
       117.76242536, 125.25245375])
```

```
min_points = 1500
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import euclidean_distances
datt=StandardScaler().fit_transform(dat)

distance=[]
for point in tqdm1(datt):
    temp = euclidean_distances(datt, point.reshape(1, -1))
    distance.append(temp[min_points])
sorted_distance = np.sort(np.array(distance))

sorted_dist = np.sort(sorted_distance.reshape(1,-1)[0])
points = [i for i in range(len(datt))]


# Draw distances(d_i) VS points(x_i) plot
plt.plot(points, sorted_dist)
plt.xlabel('Points')
plt.ylabel('dist')
plt.title('dist VS Points')
plt.grid()
plt.show()
```

```
HBox(children=(IntProgress(value=0, max=5000), HTML(value='')))
```




```
#we can see that point of inflexion is at eps=9
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=90,n_jobs=-1)
dbscan.fit(datt)
print('No of clusters: ',len(set(dbscan.labels_)))
print('Cluster are including noise i.e -1: ',set(dbscan.labels_))
```

```
No of clusters:  2
Cluster are including noise i.e -1:  {0, -1}
```

```
#ignoring -1 as it is for noise
cluster1=[]
noisecluster1=[]
for i in range(dbscan.labels_.shape[0]):
    if dbscan.labels_[i] == 0:
        cluster1.append(essays[i])
    elif dbscan.labels_[i] == -1:
        noisecluster1.append(essays[i])
```

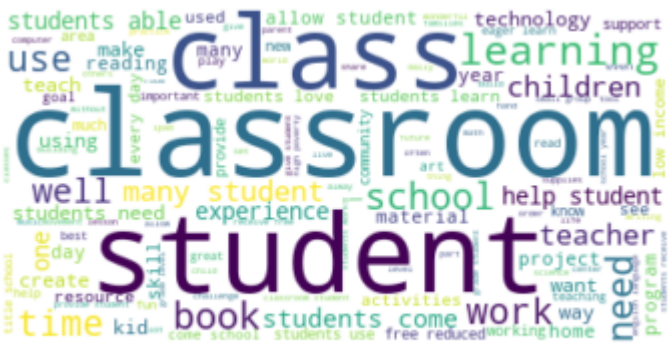
```
for i in range(3):
    print('%s\n'%(cluster1[i]))
```

```
my students diverse socioeconomic backgrounds ethnicities they hardworking want best i w

within class i great diversity learners i never school including preschool kindergarten

as educator every experience classes smart literally extra homework keep our kindergarte
```

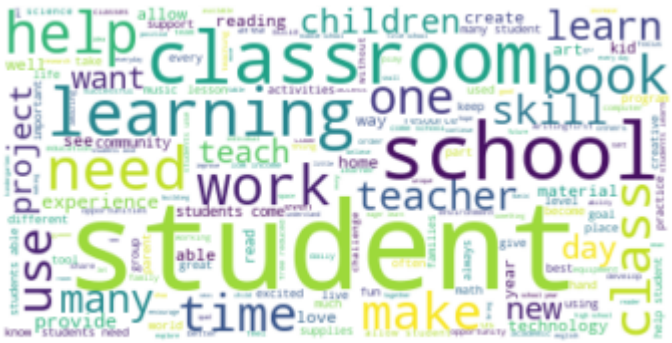
```
for i in range(3):
    print('%s\n'%(noisecluster1[i]))
```

```
 my students hungry meaning they want learn learning connects real world work improves wo

 my students english language learners non english speakers they live poverty line parent

 students come school eager learn explore they excited talk friends teacher at young age
```

```
#cluster 1
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```




```
#noise cluster 1
words=''
for i in noisecluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white").generate(words)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```




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

3. Cocnlusions

Please write down few lines of your observations on this assignment.

1.K-Means:

1.First i hyperparameter tuned K_values with 2,3,4,5,6,7,8 and got the inflection point at k=6

2.Then i trained K-Means on K_value=6

3.After training i clustered the essays into 6 seperate clusters

4.Then i plotted the word cloud

2.Agglomerative:

1.First i reduced the dimentions to 5000 and also took 5000 points same as in K-Means

2.Then i applied Agglomerative clustering on k=2

3.Then i clustered the essays into 2 seperate clusters

4.After that i plotted the wordcloud for each of the clusters

5.Then i applied Agglomerative clustering on k=5

6.Then i clustered the essays into 5 seperate clusters

7.After that i plotted the wordcloud for each of the clusters

3.DBScan:

1.First i converted the reduced sparce matrix to dense using toarray()

2.Then i transformed the data to standard scalar

3.Then i computed euclidean distance for every point to every other point and took the distance of mi

4.Obtained the best eps to be 90 from the above graph b/w dist and points

5.Then formed clusters on noise points and non-noise points

6.Printed the essays in each of the two clusters

7.Then printed the wordcloud.

Pretty Table

K-Means:

```
#prettytable for kmeans
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer","Best k"]
x.add_row(['TFIDF','6'])
print(x)
```

```
+------------+--------+
| Vectorizer | Best k |
+------------+--------+
|   TFIDF    |   6    |
+------------+--------+
```

Agglomerative:

```
#prettytable for kmeans
```

```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer","Best k"]
x.add_row(['TFIDF','2'])
x.add_row(['TFIDF','5'])
print(x)
```

```
+------------+--------+
| Vectorizer | Best k |
+------------+--------+
|   TFIDF    |   2    |
|   TFIDF    |   5    |
+------------+--------+
```

DBScan:

```
#prettytable for kmeans
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Vectorizer","Best k","Eps","Number of clusters(INCLUDING NOISE)"]
x.add_row(['TFIDF','2',90,2])
print(x)
```

```
+------------+--------+-----+--------------------------------------+
| Vectorizer | Best k | Eps | Number of clusters(INCLUDING NOISE) |
+------------+--------+-----+--------------------------------------+
|   TFIDF    |   2    |  90 |                  2                   |
+------------+--------+-----+--------------------------------------+
```

Follow link (ctrl + click)

Follow link (ctrl + click)