DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in nee volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoos

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three r

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

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a tea descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then u 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 o • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the pro • `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). **Ex**
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Examp** • `My students need hands on literacy materials`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*

Feature	Description
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** 2
`teacher_id`	A unique identifier for the teacher of the proposed project. **Exa**
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the sa

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

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Eac project:

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

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you u project:

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

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project v

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about t school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' lea

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 ar

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=9473

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

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

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

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

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

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

from tqdm import tqdm
import os


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

1.1 Reading Data

```
project_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/train_data.csv', nrows=50000)
resource_data = pd.read_csv('/content/drive/My Drive/Colab Notebooks/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 (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_sta
 '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)
```

```
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/47301924/408

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care &
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scie
            j=j.replace('The','') # if we have the words "The" we are going to replace it with ''
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math & Scie
        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/47301924/408

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

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

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

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

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

1.3 Text preprocessing

```
# 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
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL

```
#### 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)
```

```
My students are English learners that are working on English as their second or third
==================================================
The 51 fifth grade students that will cycle through my classroom this year all love l
==================================================
How do you remember your days of school? Was it in a sterile environment with plain w
==================================================
My kindergarten students have varied disabilities ranging from speech and language de
==================================================
```

```
# 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
```

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

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

```
# \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 de
```

```
#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 de
```

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'hi
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'the
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 't
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'havin
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
```

```
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during'
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'unde
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very',
            '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", 'doesn',
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "migh
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "was
            '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', ' ')
    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%|██████████| 50000/50000 [00:27<00:00, 1813.85it/s]
```

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

```
'my kindergarten students varied disabilities ranging speech language delays cognitiv
```

1.4 Preprocessing of `project_title`

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

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

```
100%|██████████| 50000/50000 [00:01<00:00, 42222.20it/s]
```

```
project_data['project_title'] = preprocessed_titles
```

```
#Preprocessing project_grade_category

#reference link: https://stackoverflow.com/questions/28986489/python-pandas-how-to-replace-a-char

project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-',
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ',
```

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

1.5.1 Vectorizing Categorical data

7 cells hidden

1.5.2 Vectorizing Text data

16 cells hidden

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.preprocessin
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.   287
# 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 devi
print(f"Mean : {price_scalar.mean_[0]}, Standard deviation : {np.sqrt(price_scalar.var_[0])}")
```

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

```
Mean : 299.33367619999996, Standard deviation : 378.20927190421384
```

```
price_standardized
```

```
array([[-0.38268146],
       [-0.00088225],
       [ 0.57512161],
       ...,
       [-0.65382764],
       [-0.52109689],
       [ 0.54492668]])
```

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)
```

```
(50000, 9)
(50000, 30)
(50000, 12211)
(50000, 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
```

```
(50000, 12251)
```

```
# 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
nltk.download()

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest students
for learning my students learn in many different ways using all of our senses and multiple intell
of techniques to help all my students succeed students in my class come from a variety of differe
for wonderful sharing of experiences and cultures including native americans our school is a cari
learners which can be seen through collaborative student project based learning in and out of the
```

```
in my class love to work with hands on materials and have many different opportunities to practic
mastered having the social skills to work cooperatively with friends is a crucial aspect of the k
montana is the perfect place to learn about agriculture and nutrition my students love to role pl
in the early childhood classroom i have had several kids ask me can we try cooking with real food
and create common core cooking lessons where we learn important math and writing concepts while c
food for snack time my students will have a grounded appreciation for the work that went into mak
of where the ingredients came from as well as how it is healthy for their bodies this project wou
nutrition and agricultural cooking recipes by having us peel our own apples to make homemade appl
and mix up healthy plants from our classroom garden in the spring we will also create our own coo
shared with families students will gain math and literature skills as well as a life long enjoyme
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
```

```
NLTK Downloader
---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> d

Download which package (l=list; x=cancel)?
  Identifier> vader_lexicon
    Downloading package vader_lexicon to /root/nltk_data...

---------------------------------------------------------------------------
    d) Download   l) List    u) Update   c) Config   h) Help   q) Quit
---------------------------------------------------------------------------
Downloader> q
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 got the b
- step 2: Choose any of the feature selection/reduction algorithms ex: selectkbest features, pretrained word 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 hierarch 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 mentioned

2. Clustering

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

```
# 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
```

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

x = project_data.drop(['project_is_approved'], axis=1)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

```
# 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
```

```
#encoding numerical features

#Price
from sklearn.preprocessing import Normalizer
normalizer = Normalizer()

normalizer.fit(x['price'].values.reshape(1, -1))

x_price_norm = normalizer.transform(x['price'].values.reshape(1, -1))

print('AFter vectorizations:')
print(x_price_norm.shape, y.shape)
```

```
AFter vectorizations:
(1, 50000) (50000,)
```

```
#Teacher_number_of_previously_posted_projects

normalizer.fit(x['teacher_number_of_previously_posted_projects'].values.reshape(1, -1))

x_previous_projects_norm = normalizer.transform(x['teacher_number_of_previously_posted_projects'])

print('After Vectorizations:')
print(x_previous_projects_norm.shape, y.shape)
```

```
After Vectorizations:
(1, 50000) (50000,)
```

```
#Encoding Categorical Features
```

```
#school_state
state_vectorizer = CountVectorizer()
state_vectorizer.fit(x['school_state'].values)

x_state_one_hot = state_vectorizer.transform(x['school_state'].values)

print("After Vectorizations:")
print(x_state_one_hot.shape, y.shape)
```

```
After Vectorizations:
(50000, 51) (50000,)
```

```
#teacher_prefix

teacher_vectorizer = CountVectorizer()
teacher_vectorizer.fit(x['teacher_prefix'].values.astype('U'))

x_teacher_one_hot = teacher_vectorizer.transform(x['teacher_prefix'].values.astype('U'))

print("After Vectorizations:")
print(x_teacher_one_hot.shape, y.shape)
```

```
After Vectorizations:
(50000, 6) (50000,)
```

```
#Project_grade_category

grade_vectorizer = CountVectorizer()
grade_vectorizer.fit(x['project_grade_category'].values)

x_grade_one_hot = grade_vectorizer.transform(x['project_grade_category'].values)

print("After Vectorizations:")
print(x_grade_one_hot.shape, y.shape)
```

```
After Vectorizations:
(50000, 4) (50000,)
```

```
#project_subject_categories


categories_vectorizer = CountVectorizer()
categories_vectorizer.fit(x['clean_categories'].values)

x_categories_one_hot = categories_vectorizer.transform(x['clean_categories'].values)

print("After Vectorizations:")
print(x_categories_one_hot.shape, y.shape)
```

```
After Vectorizations:
(50000, 9) (50000,)
```

```
#project_subject_subcategories


subcategories_vectorizer = CountVectorizer()
subcategories_vectorizer.fit(x['clean_subcategories'].values)

x_subcategories_one_hot = subcategories_vectorizer.transform(x['clean_subcategories'].values)

print("After Vectorizations:")
print(x_subcategories_one_hot.shape, y.shape)
```

```
After Vectorizations:
(50000, 30) (50000,)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

```
# 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
```

```
essay_bow_vectorizer = CountVectorizer(min_df=10, ngram_range=(1,1))
essay_bow_vectorizer.fit(x['essay'].values)

x_essay_bow = essay_bow_vectorizer.transform(x['essay'].values)

print("After Vectorizations:")
print(x_essay_bow.shape, y.shape)
```

```
After Vectorizations:
(50000, 12622) (50000,)
```

```
title_bow_vectorizer = CountVectorizer(min_df=10, ngram_range=(1,1))
title_bow_vectorizer.fit(x['project_title'].values)

x_title_bow = title_bow_vectorizer.transform(x['project_title'].values)

print("After Vectorizations:")
print(x_title_bow.shape, y.shape)
```

```
After Vectorizations:
(50000, 2039) (50000,)
```

```
#Before merging, re-shape some features. if we dont reshape, we'll get error
#Re-shaping
x_price_norm = x_price_norm.reshape(-1,1)
x_previous_projects_norm = x_previous_projects_norm.reshape(-1,1)
```

```
#merging features

from scipy.sparse import hstack
x_bow = hstack((x_price_norm, x_previous_projects_norm, x_state_one_hot, x_teacher_one_hot, x_gra
                x_categories_one_hot, x_subcategories_one_hot, x_essay_bow, x_title_bow)).tocsr()
```

2.4 Dimensionality Reduction on the selected features

```
# 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
```

```
from sklearn.feature_selection import SelectKBest, chi2

top_features = SelectKBest(chi2, k=5000).fit_transform(x_bow, y)
top_features.shape
```

```
(50000, 5000)
```

```
top_features = top_features[:30000]
top_features.shape
```

```
(30000, 5000)
```

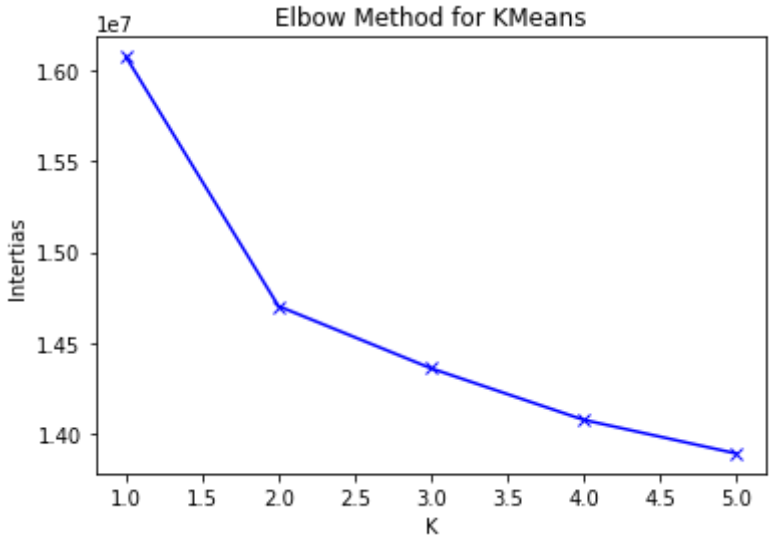
2.5 Apply Kmeans

```
# 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
```

```
# Reference https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/
```

```
from sklearn.cluster import KMeans
inertias = []

K = range(1, 6)
for k in K:
    model = KMeans(n_clusters=k).fit(top_features)
    model.fit(top_features)
    inertias.append(model.inertia_)
```

```
plt.plot(K, inertias, 'bx-')
plt.xlabel('K')
plt.ylabel('Intertias')
plt.title('Elbow Method for KMeans')
plt.show()
```



```
from sklearn.cluster import KMeans

model = KMeans(n_clusters = 2, n_jobs=-1)
model.fit(top_features)
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=2, n_init=10, n_jobs=-1, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

```
a = model.labels_
```

```
project_data = project_data[:30000]
project_data['Kmeans_labels'] = a
```

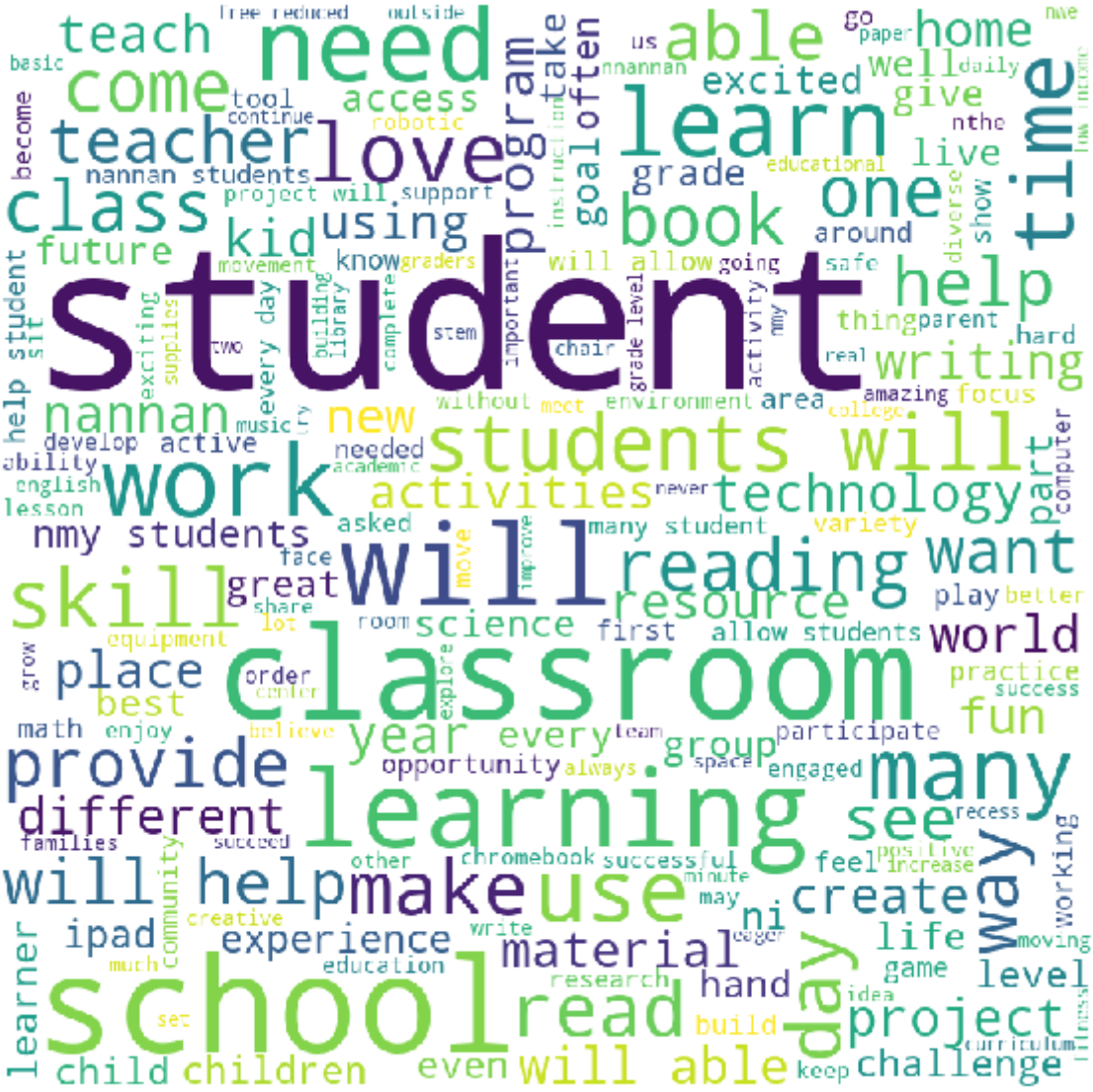
```
project_data['Kmeans_labels'].value_counts()
```

```
1    21029
0     8971
Name: Kmeans_labels, dtype: int64
```

```
#getting essays where Kmeans_labels == 0
essays_0 = project_data.loc[project_data['Kmeans_labels'] == 0]
essays_0 = essays_0['essay']
essays_0 = essays_0[:100]  #Taking only 100 essays for the wrodcloud
```

```
import nltk
from wordcloud import WordCloud, STOPWORDS
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_0:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



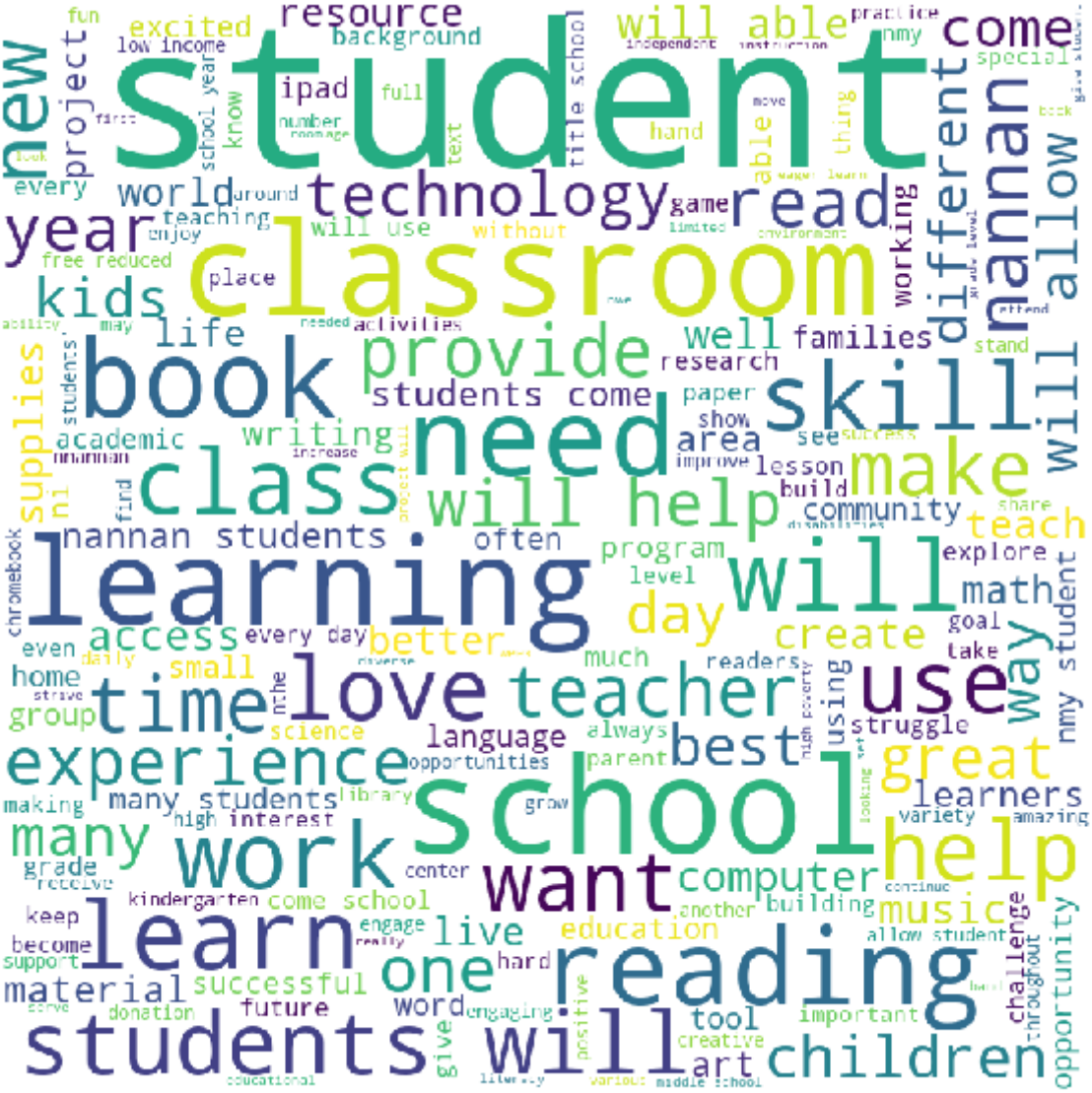

```
#Most of the points in the above cluster are about studies, school things etc.
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', ' student', 'work', 'learning', 'skill', 'use', 'teacher', 'read', 'n
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
 student
work
learning
skill
use
teacher
read
need
book
able
writing
help
love
provide
```

```
#getting essays where Kmeans_labels == 1
essays_1 = project_data.loc[project_data['Kmeans_labels'] == 1]
essays_1 = essays_1['essay']
essays_1 = essays_1[:100]  #Taking only 100 essays for the wrodcloud
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_1:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```




```
#Most of the points in the above cluster are about school, and things in school like class, learn
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', ' student', 'work', 'learning', 'skill', 'use', 'teacher', 'read', 'n
    'experience', 'year', 'kids', 'make', 'different', 'allow']
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
 student
work
learning
skill
use
teacher
read
need
book
able
writing
help
computer
provide
experience
year
kids
make
different
allow
```

2.6 Apply AgglomerativeClustering

```
# 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
```

```
from sklearn.cluster import AgglomerativeClustering
```

```
#Converting top_features into dense, because model is throwing an error to convert it into dense
top_features = top_features.toarray()
```

```
#With 2 clusters
model = AgglomerativeClustering(n_clusters=2)
model.fit(top_features)
```

```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
                        connectivity=None, distance_threshold=None,
                        linkage='ward', memory=None, n_clusters=2,
                        pooling_func='deprecated')
```

```
a = model.labels_
```

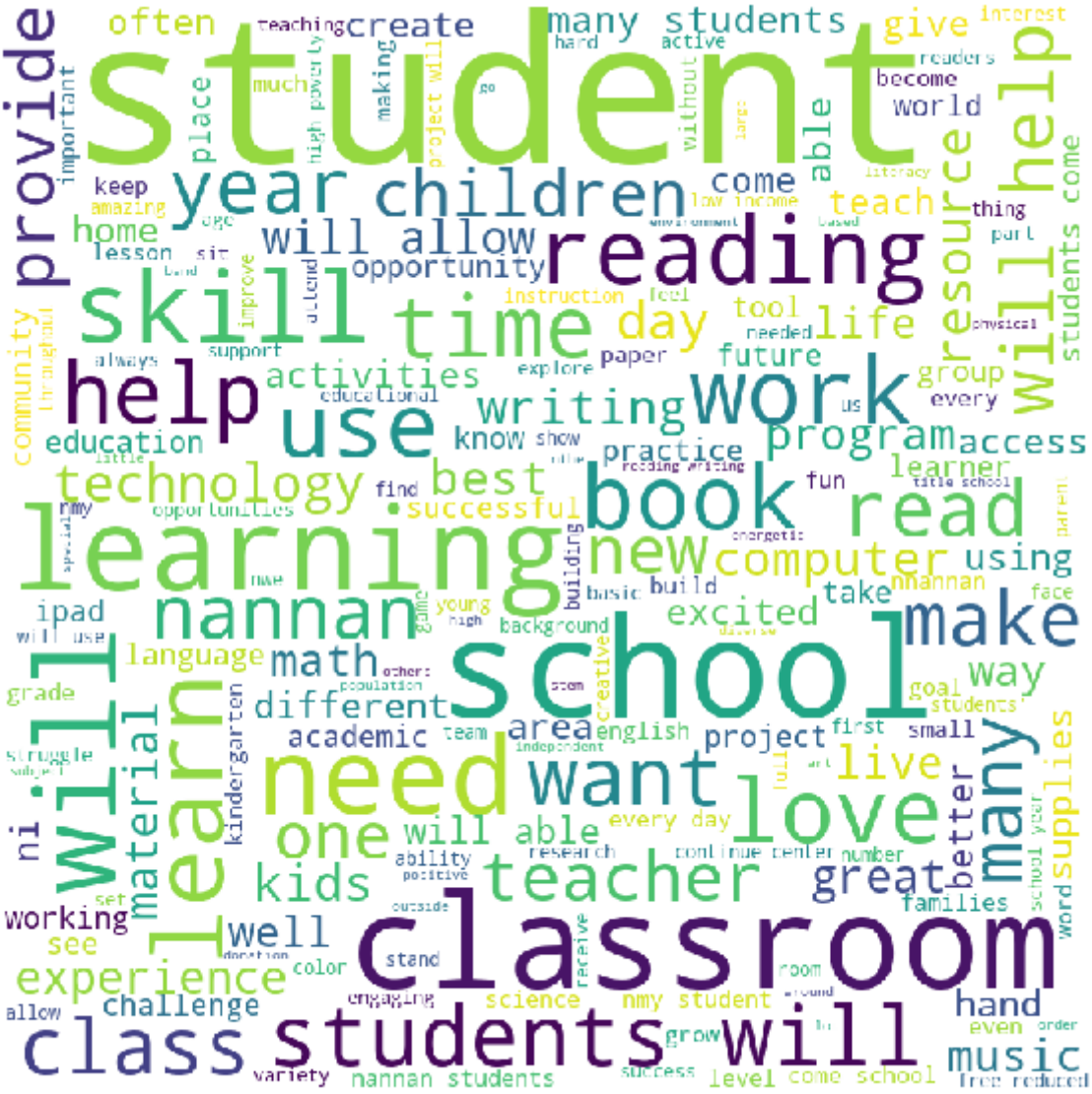
```
project_data = project_data[:30000]
project_data['Agglomerative_2clusters_labels'] = a
```

```
project_data['Agglomerative_2clusters_labels'].value_counts()
```

```
0    24647
1     5353
Name: Agglomerative_2clusters_labels, dtype: int64
```

```
#getting essays where Agglomerative_2clusters_labels == 0
essays_0 = project_data.loc[project_data['Agglomerative_2clusters_labels'] == 0]
essays_0 = essays_0['essay']
essays_0 = essays_0[:100]  #Taking only 100 essays for the wrodcloud
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_0:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



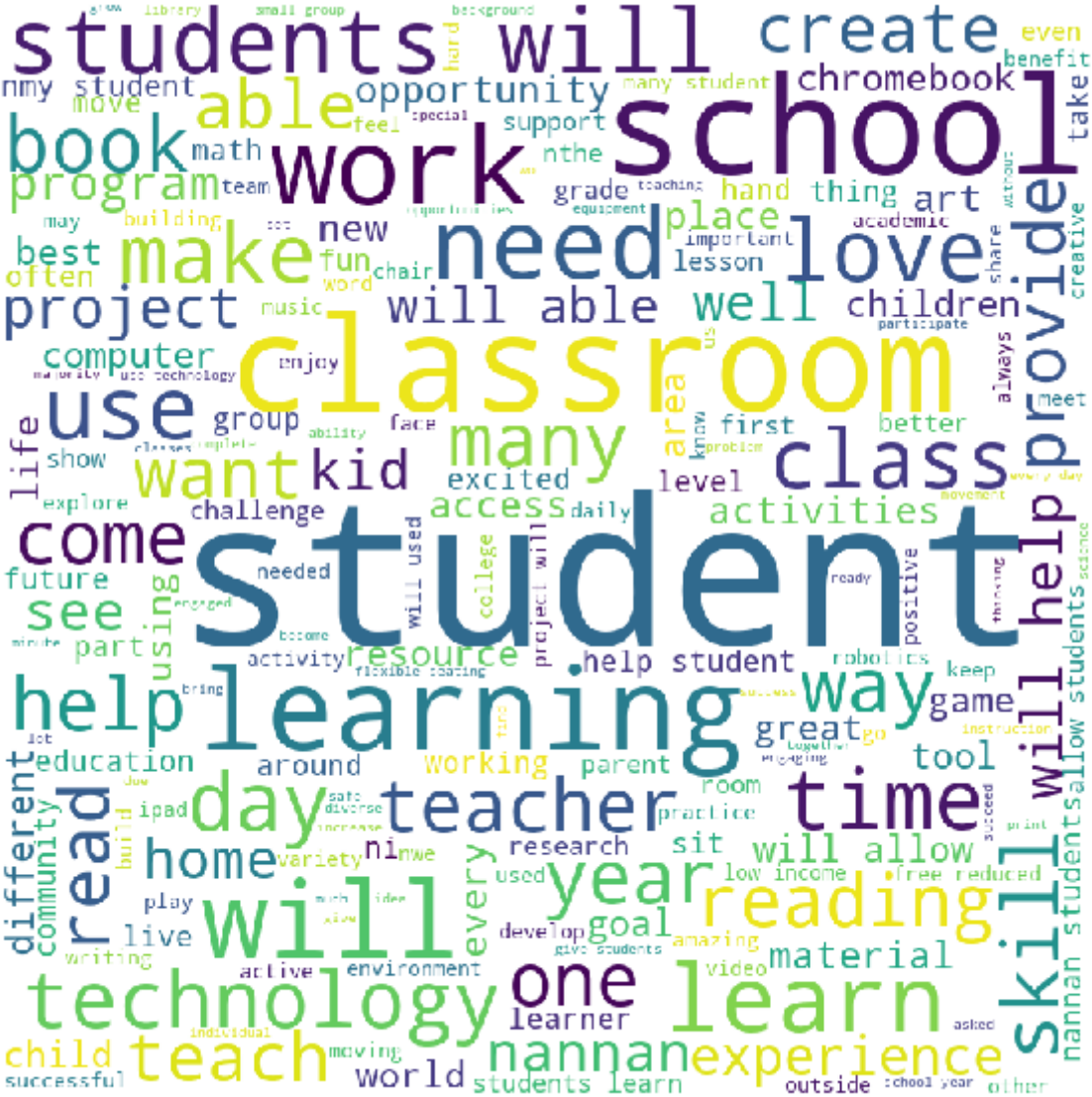

```
#Most of the points in the above cluster are about students and student life like learning, read,
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', 'year', 'learn', ' student', 'time', 'work', 'learning', 'skill', 'us
     'love', 'provide', 'technology', 'great', 'best']
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
year
learn
 student
time
work
learning
skill
use
teacher
read
need
book
able
writing
help
love
provide
technology
great
best
```

```
#getting essays where Agglomerative_2clusters_labels == 1
essays_1 = project_data.loc[project_data['Agglomerative_2clusters_labels'] == 1]
essays_1 = essays_1['essay']
essays_1 = essays_1[:100]  #Taking only 100 essays for the wrodcloud
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_1:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```




```
#Most of the points are about students and students needs in school etc
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', ' student', 'will', 'create', 'work', 'learning', 'skill', 'use', 'te
     'provide', 'make', 'project', 'computer', 'use', 'many', 'reading', 'goal']
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
 student
will
create
work
learning
skill
use
teacher
read
need
book
able
writing
help
love
provide
make
project
computer
use
many
reading
goal
```

```
#With 5 clusters
model = AgglomerativeClustering(n_clusters=5)
model.fit(top_features)
```

```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
                        connectivity=None, distance_threshold=None,
                        linkage='ward', memory=None, n_clusters=5,
                        pooling_func='deprecated')
```

```
a = model.labels_
project_data['Agglomerative_5clusters_labels'] = a
```

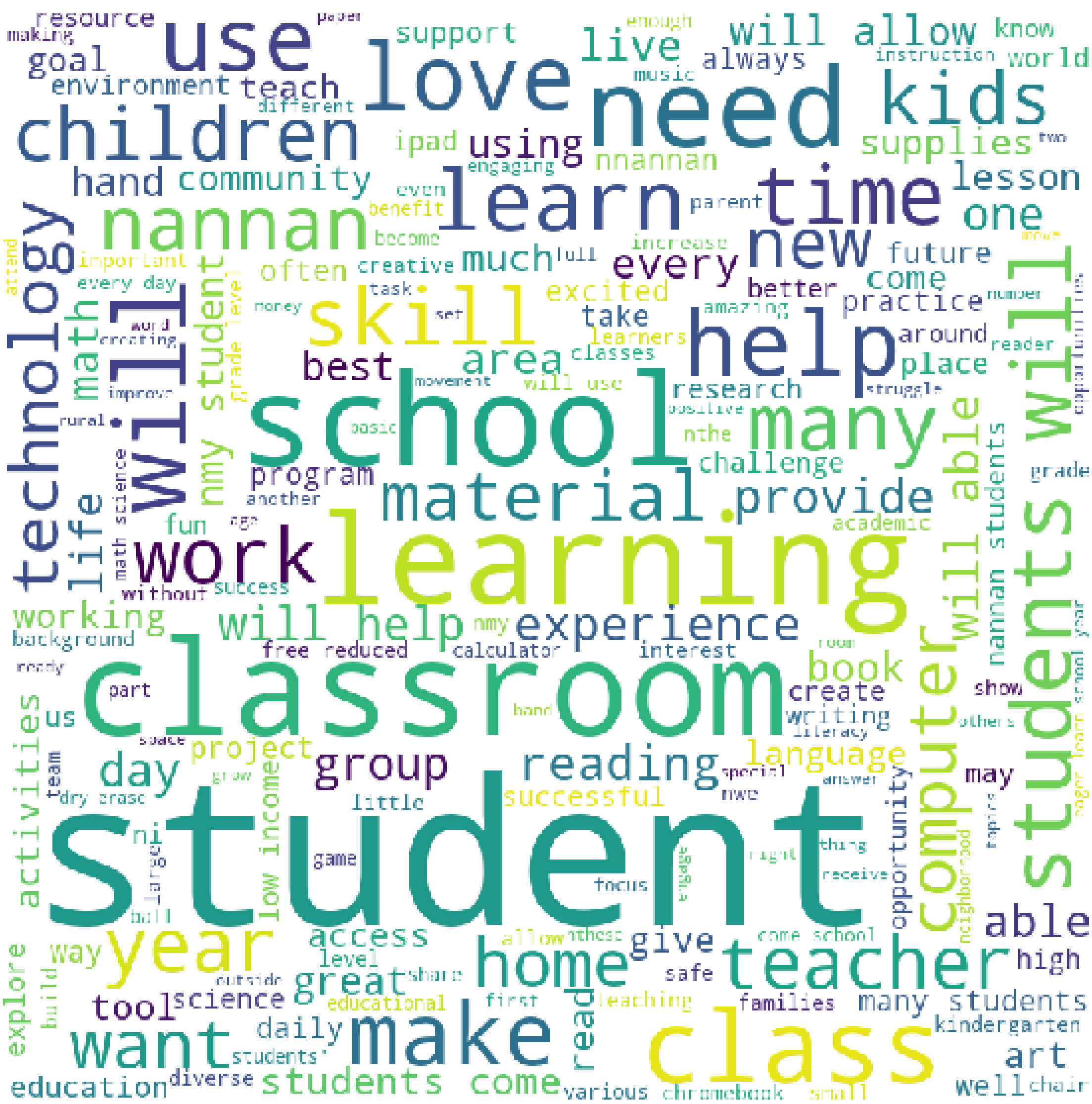
```
project_data['Agglomerative_5clusters_labels'].value_counts()
```

```
0    12608
1     7365
3     4674
4     3448
2     1905
Name: Agglomerative_5clusters_labels, dtype: int64
```

```
#getting essays where Agglomerative_2clusters_labels == 0
essays_0 = project_data.loc[project_data['Agglomerative_5clusters_labels'] == 0]
essays_0 = essays_0['essay']
essays_0 = essays_0[:100]  #Taking only 100 essays for the wrodcloud
```

```
comment_words = ' '
```

```python
stopwords = set(STOPWORDS)

for val in essays_0:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



```
#Most of the points in the above cluster are about school things
```

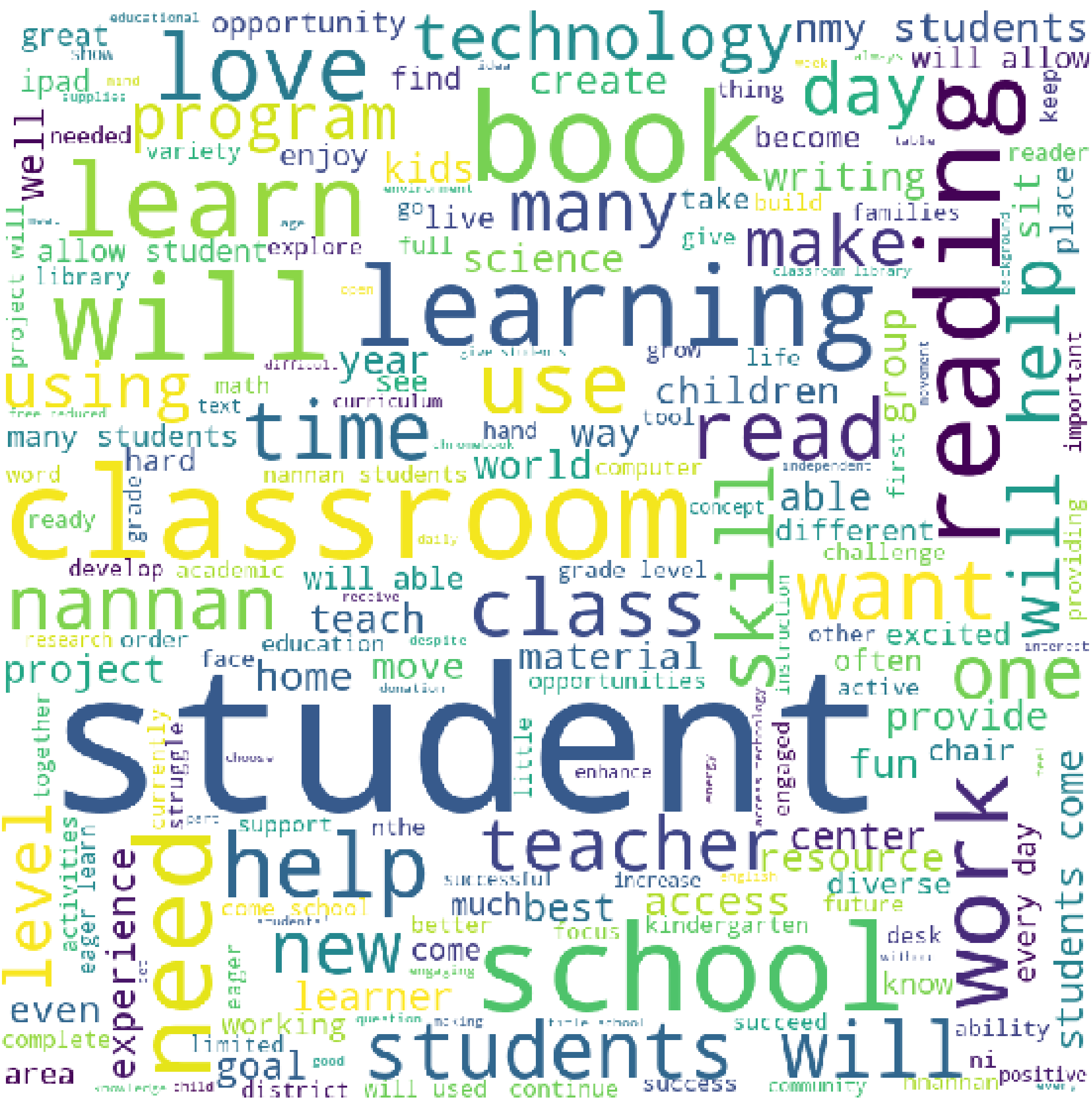
```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', ' student', 'technology', 'create', 'work', 'learning', 'skill', 'tea
     'love', 'computer', 'time', 'project', 'material', 'use', 'many', 'reading', 'kids', 'childr
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
 student
technology
create
work
learning
skill
teacher
read
need
book
community
writing
help
love
computer
time
project
material
use
many
reading
kids
children
```

```
#getting essays where Agglomerative_2clusters_labels == 1
essays_1 = project_data.loc[project_data['Agglomerative_5clusters_labels'] == 1]
essays_1 = essays_1['essay']
essays_1 = essays_1[:100]  #Taking only 100 essays for the wrodcloud
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_1:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '
```

```
wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



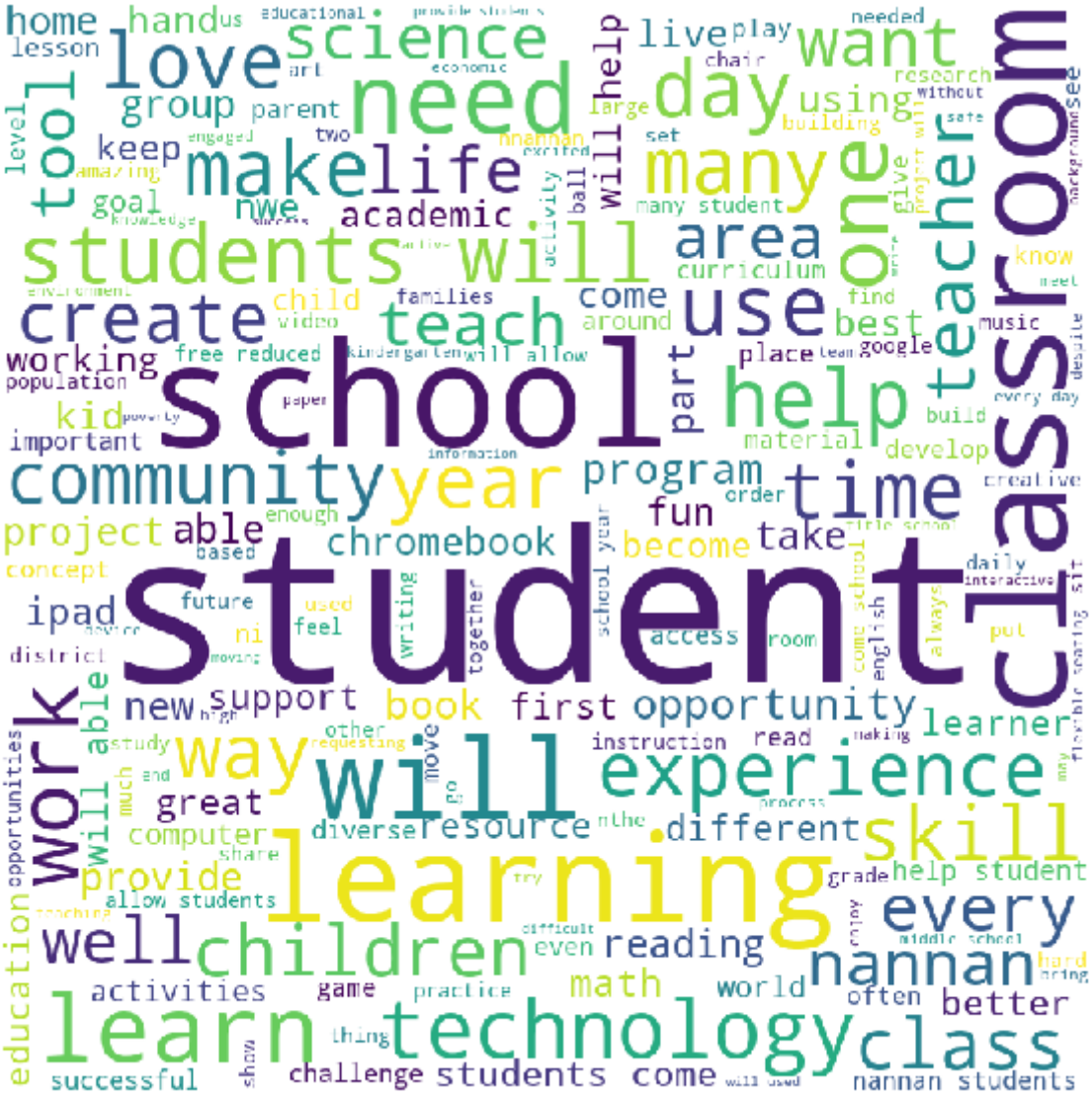

```
#Most of the points in the above cluster are about student, school things
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', ' student', 'learning', 'skill', 'writing', 'help', 'love', 'technolc
     'project', 'reading', 'goal']
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
 student
learning
skill
writing
help
love
technology
program
book
time
want
skill
help
needproject
reading
goal
```

```
#getting essays where Agglomerative_2clusters_labels == 2
essays_2 = project_data.loc[project_data['Agglomerative_5clusters_labels'] == 2]
essays_2 = essays_2['essay']
essays_2 = essays_2[:100]  #Taking only 100 essays for the wrodcloud
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_2:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



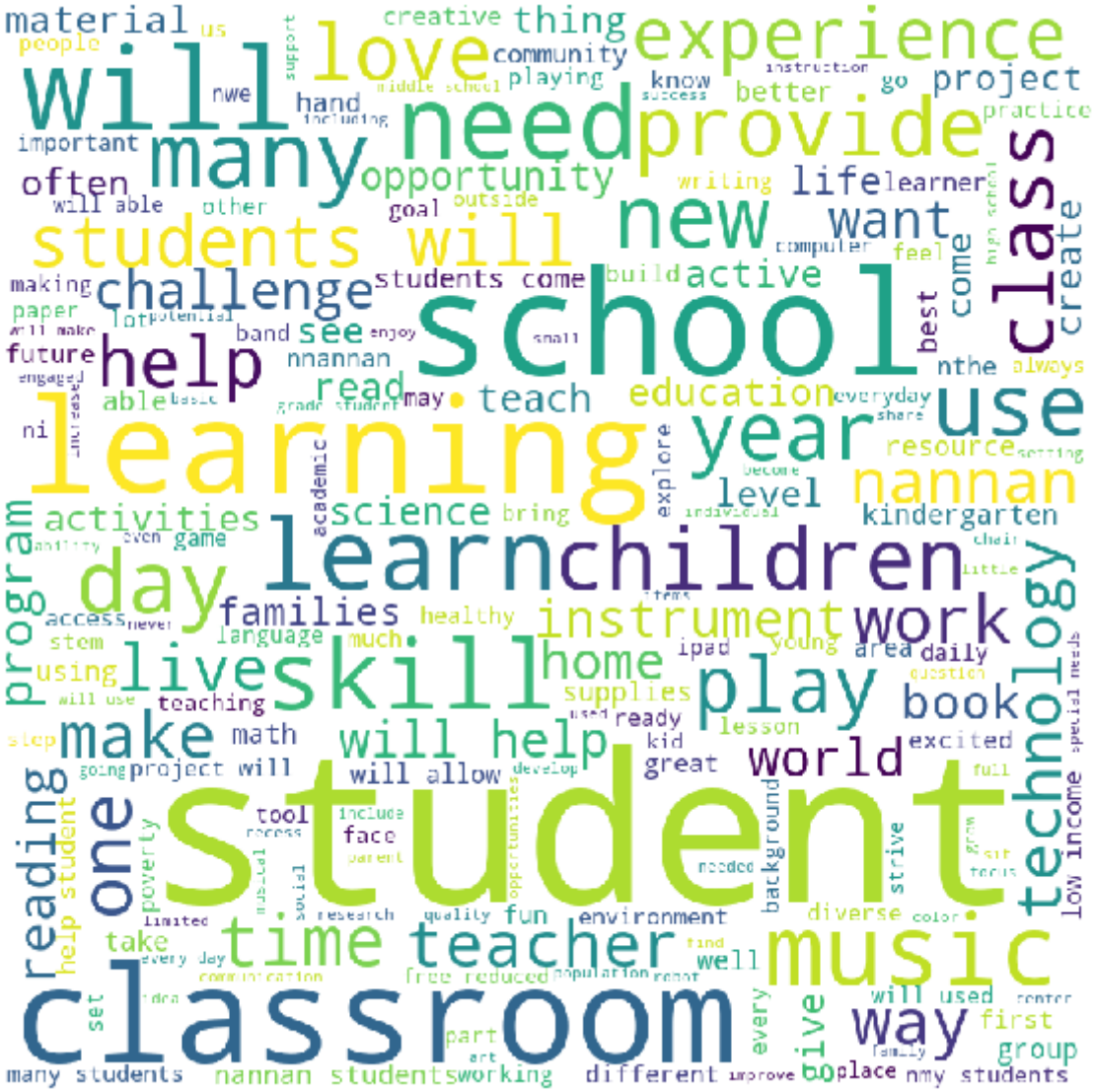

```
#Most of the points in the above cluster are about school, classroom, students
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', ' student', 'learning', 'skill', 'writing', 'help', 'love', 'science'
     'reading', 'goal', 'experience', 'opportunity', 'technology', 'provide']
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
 student
learning
skill
writing
help
love
science
life
create
academic
communtiy
year
program
reading
goal
experience
opportunity
technology
provide
```

```
#getting essays where Agglomerative_2clusters_labels == 3
essays_3 = project_data.loc[project_data['Agglomerative_5clusters_labels'] == 3]
essays_3 = essays_3['essay']
essays_3 = essays_3[:100]  #Taking only 100 essays for the wrodcloud
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_3:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```



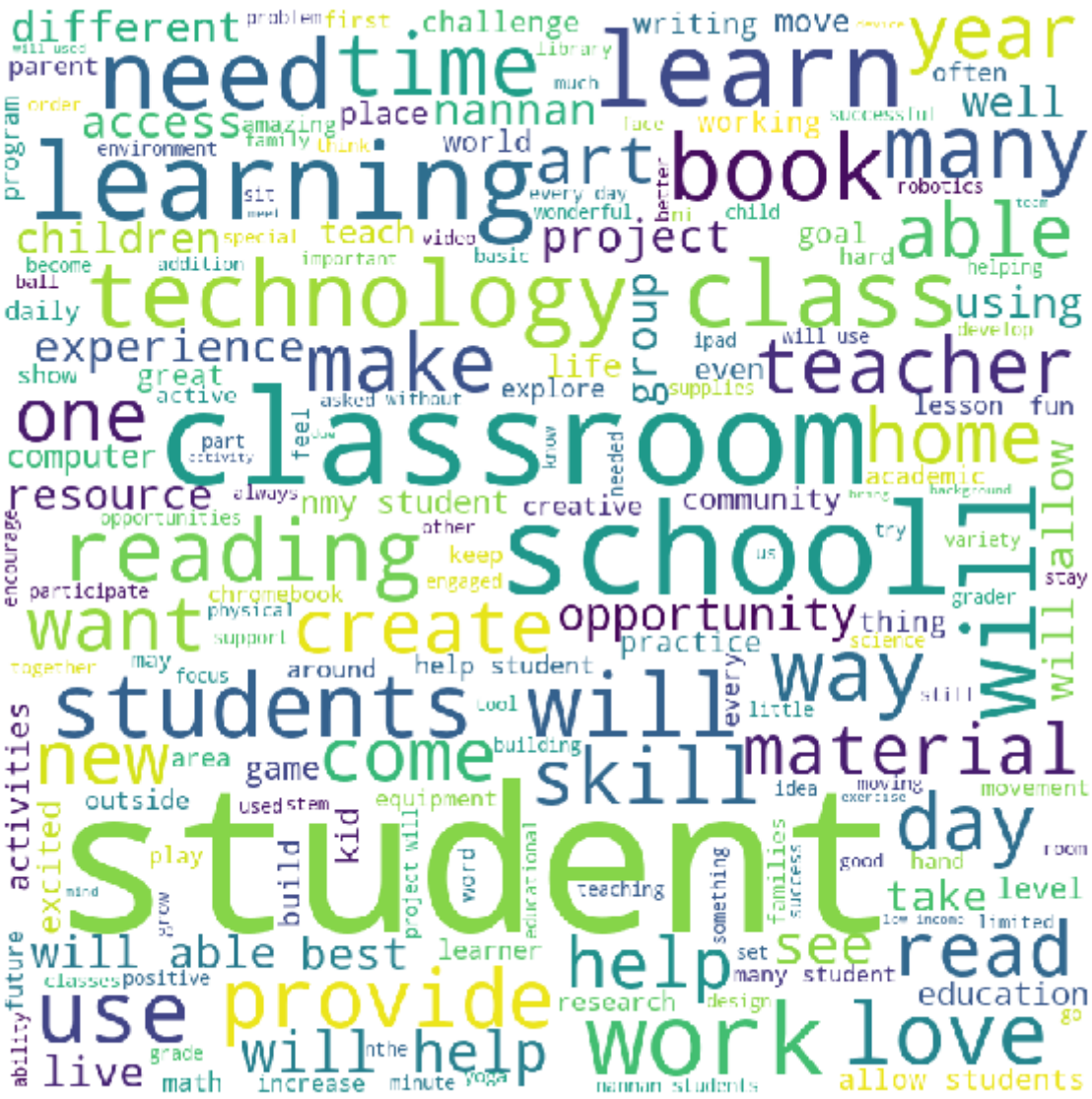

```
#Most of the points in the above cluster are aout school, learning, love, help etc
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', ' student', 'learning', 'skill', 'writing', 'help', 'love', 'experien
     'reading', 'goal', 'play', 'education']
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
 student
learning
skill
writing
help
love
experience
need
help
challenge
create
instrument
wayreading
goal
play
education
```

```
#getting essays where Agglomerative_2clusters_labels == 4
essays_4 = project_data.loc[project_data['Agglomerative_5clusters_labels'] == 4]
essays_4 = essays_4['essay']
essays_4 = essays_4[:100]  #Taking only 100 essays for the wrodcloud
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_4:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```




```
#Most of the points in the above cluster are about learning, technologies, experience etc.
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', ' student', 'learning', 'skill', 'writing', 'help', 'love', 'need', '
     'reading', 'goal', 'help', 'opportunity', 'home', 'computer', 'will', 'way', 'reading', 'wa
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
 student
learning
skill
writing
help
love
need
year
book
many
project
material
create
provide
reading
goal
help
opportunity
home
computer
will
way
reading
want
use
```

2.7 Apply DBSCAN

```
# 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
```

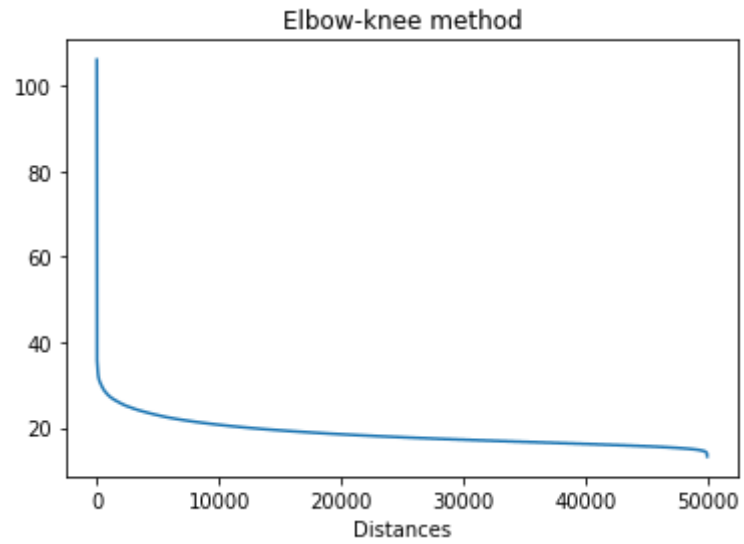
```
from sklearn.feature_selection import SelectKBest, chi2

features = SelectKBest(chi2, k=1500).fit_transform(x_bow, y)
features = features[:30000]
features.shape
```

```
(30000, 1500)
```

```
#https://scikit-learn.org/stable/modules/neighbors.html
from sklearn.neighbors import NearestNeighbors

ns = 2*1500
nbrs = NearestNeighbors(n_neighbors=ns).fit(features)
distances, indices = nbrs.kneighbors(features)
```

```
distancesDec = sorted(distances[:, ns-1], reverse=True)
```

```
import matplotlib.pyplot as plt

plt.plot(distancesDec)
plt.xlabel('Distances')
plt.title('Elbow-knee method')
plt.show()
```




```
from sklearn.cluster import DBSCAN

model = DBSCAN(eps=25)
model.fit(features)
```

```
DBSCAN(algorithm='auto', eps=25, leaf_size=30, metric='euclidean',
       metric_params=None, min_samples=5, n_jobs=None, p=None)
```

```
a = model.labels_
```

```
project_data = project_data[:30000]
project_data['DBSCAN_labels'] = a
```

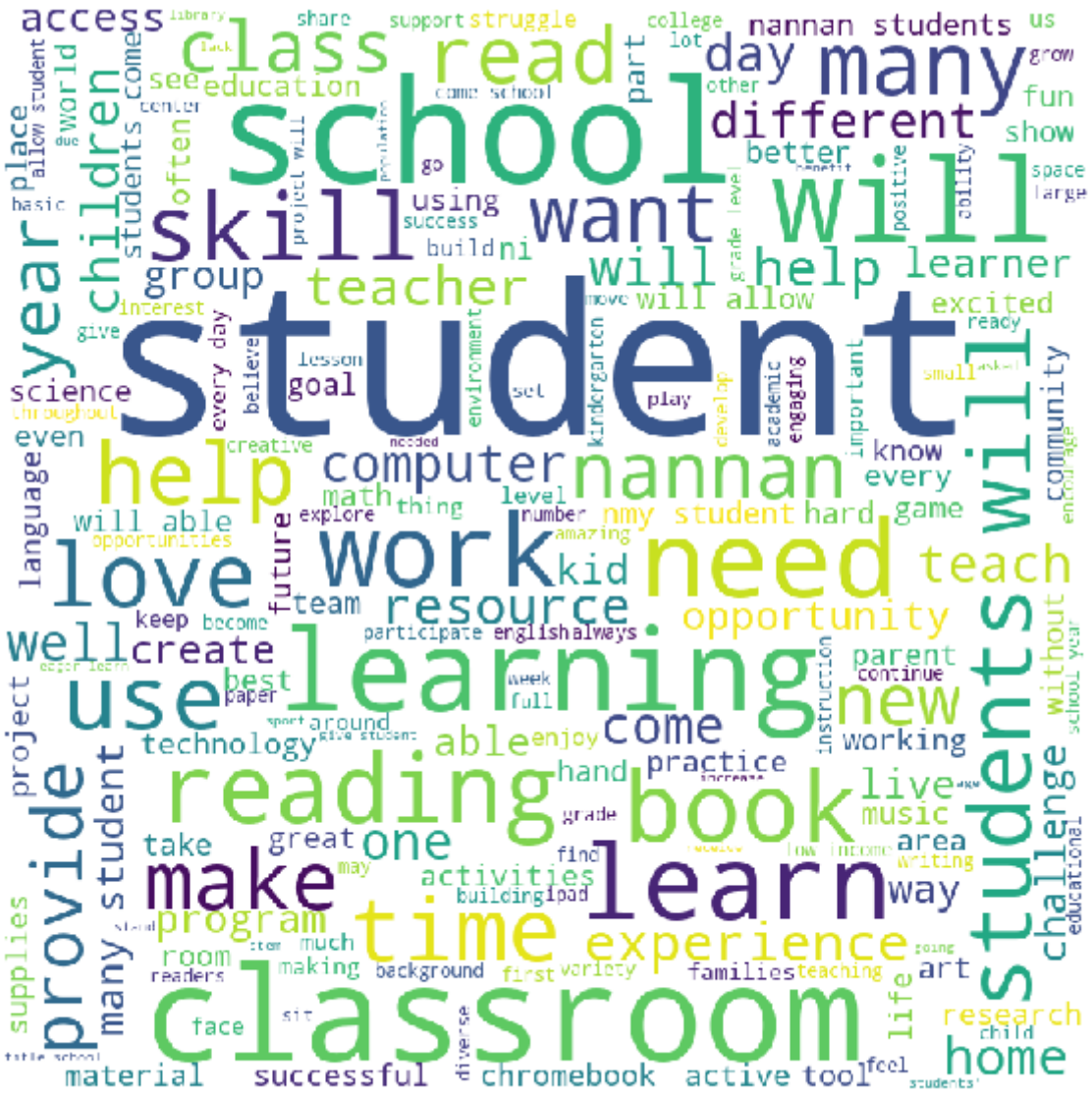
```
project_data['DBSCAN_labels'].value_counts()
```

```
 0    29953
-1       47
Name: DBSCAN_labels, dtype: int64
```

```
essays_0 = project_data.loc[project_data['DBSCAN_labels'] == 0]
essays_0 = essays_0['essay']
essays_0 = essays_0[:100]  #Taking only 100 essays for the wrodcloud
```

```
import nltk
from wordcloud import WordCloud, STOPWORDS
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_0:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```




```
#Most of the points in the above cluster are about students and school
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'school', ' student', 'learning', 'skill', 'want', 'many', 'different', 'comput
     'reading', 'goal', 'time', 'experience', 'live', 'opportunity']
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
school
 student
learning
skill
want
many
different
computer
need
writing
help
love
use
book
learnreading
goal
time
experience
live
opportunity
```

```
essays_1 = project_data.loc[project_data['DBSCAN_labels'] == -1]
essays_1 = essays_1['essay']
```

```
comment_words = ' '
stopwords = set(STOPWORDS)

for val in essays_1:

    # typecaste each val to string
    val = str(val)

    # split the value
    tokens = val.split()

    # Converts each token into lowercase
    for i in range(len(tokens)):
        tokens[i] = tokens[i].lower()

    for words in tokens:
        comment_words = comment_words + words + ' '


wordcloud = WordCloud(width = 800, height = 800,
                background_color ='white',
                stopwords = stopwords,
                min_font_size = 10).generate(comment_words)

# plot the WordCloud image
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```


```
#Most of the points in the above cluster are about student, learning, skils, time etc
```

```
#Manually printing a few top words from the above wordcloud plot

a = ['classroom', 'visual', 'learners', 'hands', 'project', 'school', ' student', 'learning', 'sk
     'reading', 'goal', 'access', 'movement', 'way']
print('Some top words:')
for i in range(len(a)):
  print(a[i])
```

```
Some top words:
classroom
visual
learners
hands
project
school
 student
learning
skill
writing
help
love
technology
home
activities
reading
goal
access
movement
way
```

3. Cocnlusions

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

```
#I think this assignment is very useful because we implemented all three most important clusterin
#In this assignment I've used some new techniques like SelectKBest, elbow plot which I haven't us
#These clustering algorithms are computationally expensive. They're taking so much of time even w
```