Donors Choose

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

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

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

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

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

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** `p036502`
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. One of the following enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`
`project_subject_categories`	One or more (comma-separated) subject categories for the project from the following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal code (https://en.wikipedia.org/wiki/List_of_U.S._state_abbreviations#Postal_codes)). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!</code`

Feature	Description
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Example:** `2016-04-28 12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed project. **Example:** `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by the same teacher. **Example:** `2`

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

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

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

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

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

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

Notes on the Essay Data

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

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

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

- __project_essay_1:__ "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- __project_essay_2:__ "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

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

In [1]:

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

import random
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from wordcloud import WordCloud, STOPWORDS
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import Normalizer
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 sklearn.model_selection import train_test_split
from scipy.sparse import hstack
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
import scipy
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

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

1. Reading the Data

In [2]:

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

In [3]:
```
## Check the shape and attributes of the project data
print("Number of data points in project train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in project train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:
```
## Check the shape and attributes of the resource data
print("Number of data points in resource train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

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

Out[4]:

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

1.1 Preprocessing Categorical Features: project_grade_category

In [5]:
```
print("Project grade" ,project_data['project_grade_category'].value_counts(dropna=False))
## visulaize how project grade looks like
print('-'*50)
print(project_data['project_grade_category'].values[1000])
print(project_data['project_grade_category'].values[1500])
print('There is no nan values for this feature ' )
```

```
Project grade Grades PreK-2    44225
Grades 3-5       37137
Grades 6-8       16923
Grades 9-12      10963
Name: project_grade_category, dtype: int64
--------------------------------------------------
Grades 3-5
Grades PreK-2
There is no nan values for this feature
```

In [6]:
```
# https://stackoverflow.com/questions/36383821/pandas-dataframe-apply-function-to-column-strings-based-on-other-column-value
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('Grades ','')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace(' ','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.replace('-','_')
project_data['project_grade_category'] = project_data['project_grade_category'].str.lower()
project_data['project_grade_category'].value_counts()
```

Out[6]:
```
prek_2    44225
3_5       37137
6_8       16923
9_12      10963
Name: project_grade_category, dtype: int64
```

1.2 Preprocessing Categorical Features: project_subject_category

In [7]:
```
categories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/47301924/4084039
# reference from course material : reference_EDA.ipynb
# 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

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



project_data['clean_categories'] = _process_cat_subcat(categories)
project_data.drop(['project_subject_categories'], axis=1, inplace=True)
project_data.head(2)

### maintain a dict that
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 Categorical Features: project_subject_category

In [8]:
```
sub_categories = list(project_data['project_subject_subcategories'].values)
project_data['clean_subcategories'] = _process_cat_subcat(sub_categories)
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]))
```

1.4 Preprocessing Categorical Features: school_state

In [9]:
```python
project_data['school_state'].value_counts()
## Convert it to Lower
project_data['school_state'] = project_data['school_state'].str.lower()
print(project_data['school_state'].value_counts(dropna=False))
print('No nan values in this feature')
```

```
ca    15388
tx     7396
ny     7318
fl     6185
nc     5091
il     4350
ga     3963
sc     3936
mi     3161
pa     3109
in     2620
mo     2576
oh     2467
la     2394
ma     2389
wa     2334
ok     2276
nj     2237
az     2147
va     2045
wi     1827
al     1762
ut     1731
tn     1688
ct     1663
md     1514
nv     1367
ms     1323
ky     1304
or     1242
mn     1208
co     1111
ar     1049
id      693
ia      666
ks      634
nm      557
dc      516
hi      507
me      505
wv      503
nh      348
ak      345
de      343
ne      309
sd      300
ri      285
mt      245
nd      143
wy       98
vt       80
Name: school_state, dtype: int64
No nan values in this feature
```

1.5 Preprocessing Categorical Features: Teacher_prefix

In [10]:
```
print(project_data['teacher_prefix'].value_counts(dropna=False))
# try to remove the dots from the teacher prefix and replace nan with mrs.
project_data['teacher_prefix']=project_data['teacher_prefix'].fillna('Mrs.')
project_data['teacher_prefix']=project_data['teacher_prefix'].str.replace('.','')
project_data['teacher_prefix']=project_data['teacher_prefix'].str.lower()
project_data['teacher_prefix']=project_data['teacher_prefix'].str.strip()
```

```
Mrs.       57269
Ms.        38955
Mr.        10648
Teacher     2360
Dr.           13
NaN            3
Name: teacher_prefix, dtype: int64
```

1.6 Combining all the essays

In [11]:
```
print('Number of nan values in essay1 is ' ,len(project_data[project_data["project_essay_1"].isna()==True]))
print('Number of nan values in essay2 is ' ,len(project_data[project_data["project_essay_2"].isna()==True]))
print('Number of nan values in essay3 is ' ,len(project_data[project_data["project_essay_3"].isna()==True]))
print('Number of nan values in essay4 is ' ,len(project_data[project_data["project_essay_4"].isna()==True]))
```

```
Number of nan values in essay1 is  0
Number of nan values in essay2 is  0
Number of nan values in essay3 is  105490
Number of nan values in essay4 is  105490
```

In [12]:
```
# 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)
```

1.7. Preprocessing Numerical Values: price

In [13]:
```
## calculate the overall count of resources and the total price for each project id
price_data=resource_data.groupby('id',as_index=False).agg({'price':'sum','quantity':'sum' })
##merge into the project_Data
project_data = pd.merge(project_data,price_data,on='id',how='left')
```

1.8 Preprocessing Text Features: project_title , essay

In [14]:

```python
# https://stackoverflow.com/a/47091490/4084039
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", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]

print("printing some random reviews")
print(9, project_data['project_title'].values[9])
print(34, project_data['project_title'].values[34])
print(147, project_data['project_title'].values[147])
```

```
printing some random reviews
9 Just For the Love of Reading--\r\nPure Pleasure
34 \"Have A Ball!!!\"
147 Who needs a Chromebook?\r\nWE DO!!
```

In [15]:
```
# Combining all the above statements
# stemming the words
from nltk.stem import SnowballStemmer
sno=SnowballStemmer('english')
def preprocess_text(text_data):
    preprocessed_text = []
    # tqdm is for printing the status bar
    for sentance in tqdm(text_data):
        sent = decontracted(sentance)
        sent = sent.replace('\\r', ' ')
        sent = sent.replace('\\n', ' ')
        sent = sent.replace('\\"', ' ')
        sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
        # https://gist.github.com/sebleier/554280
        sent = ' '.join(sno.stem(e) for e in sent.split() if e.lower() not in stopwords)
        preprocessed_text.append(sent.lower().strip())
    return preprocessed_text
```

In [16]:
```
preprocessed_titles = preprocess_text(project_data['project_title'].values)
#merge the column in the project_data
project_data['processed_title']=preprocessed_titles
```

```
100%|██████████████████████████████████████████| 109248/109248 [00:17<00:00, 6126.80it/s]
```

In [17]:
```
print("printing some random reviews")
print(9, preprocessed_titles[9])
print(34, preprocessed_titles[34])
print(147, preprocessed_titles[147])
```

```
printing some random reviews
9 love read pure pleasur
34 ball
147 need chromebook
```

In [18]:
```
print("printing some random reviews")
print(9, project_data['project_title'].values[9])
print(34, project_data['project_title'].values[34])
print(147, project_data['project_title'].values[147])
preprocessed_essays = preprocess_text(project_data['essay'].values)
```

```
printing some random reviews
9 Just For the Love of Reading--\r\nPure Pleasure
34 \"Have A Ball!!!\"
147 Who needs a Chromebook?\r\nWE DO!!
```

```
100%|██████████████████████████████████████████| 109248/109248 [10:11<00:00, 178.52it/s]
```

In [19]:

```
print("printing some random essay")
print(9, preprocessed_essays[9])
print('-'*50)
print(34, preprocessed_essays[34])
print('-'*50)
print(147, preprocessed_essays[147])

#merge the column in the project_data
project_data['processed_essay']=preprocessed_essays
```

```
printing some random essay
9 95 student free reduc lunch homeless despit come school eager learn student inquisit eager learner embrac challeng not great book resourc everi day mani not afford opportun e
ngag big color page book regular basi home not travel public librari duti teacher provid student opportun succeed everi aspect life read fundament student read book boost compr
ehens skill book use read aloud partner read independ read engag read build love read read pure enjoy introduc new author well old favorit want student readi 21st centuri know
pleasur hold good hard back book hand noth like good book read student soar read consider generous fund contribut help build stamina prepar 3rd grade thank much read propos nan
nan
--------------------------------------------------
34 student main come extrem low incom famili major come home parent work full time student school 7 30 6 00 pm 2 30 6 00 pm school program receiv free reduc meal breakfast lunc
h want student feel comfort classroom home mani student take multipl role home well school sometim caretak younger sibl cook babysitt academ friend develop go becom adult consi
d essenti part job model help other gain knowledg posit manner result communiti student love help outsid classroom consist look opportun support learn kind help way excit exper
i altern seat classroom school year studi shown give student option sit classroom increas focus well motiv allow student choic classroom abl explor creat welcom environ altern
classroom seat experi frequent recent year believ along mani other everi child learn differ not appli multipl memor paper written appli space ask work student past ask work lib
rari work carpet answer alway long learn work wherev want yoga ball lap desk abl increas option seat classroom expand imagin space nannan
--------------------------------------------------
147 student eager learn make mark world come titl 1 school need extra love fourth grade student high poverti area still come school everi day get educ tri make fun educ get sch
ool creat care environ student bloom deserv best thank request 1 chromebook access onlin intervent differenti instruct get extra practic chromebook use supplement ela math inst
ruct student play ela math game engag fun well particip assign onlin turn help student improv skill chromebook classroom would not allow student use program pace would ensur st
udent get adequ time use program onlin program especi benefici student special need abl work level well challeng differ materi make student confid abil chromebook would allow s
tudent daili access comput increas comput skill chang live better becom success school access technolog classroom would help bridg achiev gap nannan
```

3. VECTORIZING DATA

3.1 One hot encoding on Categorical : (categories,subcategories,schoolstate,teacher_prefix,projectgrade)

In [20]:

```
## remove redundant columns
project_data.drop(columns=['Unnamed: 0','project_essay_1','project_essay_2','project_essay_3','project_essay_4'],axis=1,inplace=True)
```

In [21]:
```python
X_train = project_data
y_train = project_data['project_is_approved']
X_train.drop(columns=['project_is_approved'],axis=1)
```

							students any tea...				could ask come...
...	...	...	...	...	...	...	...	...	...	...	...
-01-10 14:08:28	prek_2	***Multi-Sensory Classroom Wish!***	My students need to have a multi sensory learn...	4	Literacy_Language	Literature_Writing	My students are an amazing group of kind heart...	747.00	3	multi sensori classroom wish	student amaz group kind heart love kindergarte...
-07-26 22:43:52	prek_2	Make Learning Fun in Grade One!	My students need access to a fun, learning and...	0	Literacy_Language History_Civics	Literature_Writing SocialSciences	Creating an Interactive Learning Environment! ...	300.18	14	make learn fun grade one	creat interact learn environ help excit proud ...
-09-18 13:15:13	6_8	Hooking Young Readers with Engaging Books	My students need engaging books with topics th...	3	Literacy_Language	Literacy	Do you remember middle school? I am sure tons...	121.59	14	hook young reader engag book	rememb middl school sure ton adject pop mind m...
							Most of the				

In [22]:
```python
# we use count vectorizer to convert the values into one hot vectors
## clean categories

cat_vectorize = CountVectorizer(lowercase=False, binary=True)
cat_vectorize.fit(X_train['clean_categories'].values)

train_categories = cat_vectorize.transform(X_train['clean_categories'].values)

print(cat_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",train_categories.shape)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language', 'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of matrix of Train data after one hot encoding  (109248, 9)
```

In [23]:
```python
# we use count vectorizer to convert the values into one hot vectors
## clean subcategories

subcat_vectorize = CountVectorizer(lowercase=False, binary=True)
subcat_vectorize.fit(X_train['clean_subcategories'].values)

train_subcategories = subcat_vectorize.transform(X_train['clean_subcategories'].values)

print(subcat_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",train_subcategories.shape)

```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government', 'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics', 'EnvironmentalSc
ience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness', 'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writin
g', 'Mathematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'SocialSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
Shape of matrix of Train data after one hot encoding  (109248, 30)
```

In [24]:

```python
# we use count vectorizer to convert the values into one hot vectors
##  school state


sklstate_vectorize = CountVectorizer(lowercase=False, binary=True)
sklstate_vectorize.fit(X_train['school_state'].values)

sklstate_train = sklstate_vectorize.transform(X_train['school_state'].values)

print(sklstate_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",sklstate_train.shape)

```

```
['ak', 'al', 'ar', 'az', 'ca', 'co', 'ct', 'dc', 'de', 'fl', 'ga', 'hi', 'ia', 'id', 'il', 'in', 'ks', 'ky', 'la', 'ma', 'md', 'me', 'mi', 'mn', 'mo', 'ms', 'mt', 'nc', 'nd',
'ne', 'nh', 'nj', 'nm', 'nv', 'ny', 'oh', 'ok', 'or', 'pa', 'ri', 'sc', 'sd', 'tn', 'tx', 'ut', 'va', 'vt', 'wa', 'wi', 'wv', 'wy']
Shape of matrix of Train data after one hot encoding  (109248, 51)
```

In [25]:

```python
# we use count vectorizer to convert the values into one hot vectors
## teacher_prefix

teacher_prefix_vectorize = CountVectorizer(lowercase=False, binary=True)
teacher_prefix_vectorize.fit(X_train['teacher_prefix'].values)

teacher_prefix_train = teacher_prefix_vectorize.transform(X_train['teacher_prefix'].values)

print(teacher_prefix_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",teacher_prefix_train.shape)
```

```
['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix of Train data after one hot encoding  (109248, 5)
```

In [26]:

```python
# we use count vectorizer to convert the values into one hot vectors
## project_grade

proj_grade_vectorize = CountVectorizer(lowercase=False, binary=True)
proj_grade_vectorize.fit(X_train['project_grade_category'].values)

proj_grade_train = proj_grade_vectorize.transform(X_train['project_grade_category'].values)

print(proj_grade_vectorize.get_feature_names())
print("Shape of matrix of Train data after one hot encoding ",proj_grade_train.shape)
```

```
['3_5', '6_8', '9_12', 'prek_2']
Shape of matrix of Train data after one hot encoding  (109248, 4)
```

3.2 Vectorizing Text data

3.2.1 BOW on Essay data

In [27]:
```
 ##Considering the words that appeared in atleast 10 documents

bow_essay = CountVectorizer(min_df=10,max_features=5000)
bow_essay.fit(X_train['processed_essay'])

bow_essay_train = bow_essay.transform(X_train['processed_essay'])

print("Shape of matrix after one hot encoding ",bow_essay_train.shape)

```

```
Shape of matrix after one hot encoding  (109248, 5000)
```

3.2.2 BOW on Title data

In [28]:
```
 ##Considering the words that appeared in atleast 10 documents

bow_title = CountVectorizer(min_df=10,max_features=5000)
bow_title.fit(X_train['processed_title'])

bow_title_train = bow_title.transform(X_train['processed_title'])

print("Shape of matrix after one hot encoding ",bow_title_train.shape)


```

```
Shape of matrix after one hot encoding  (109248, 2455)
```

4. Vectorizing Numerical Features

4.1 Price

In [29]:
```
normalizer = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X_train['price'].values.reshape(1,-1))

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


## reshaping
X_train_price_norm=X_train_price_norm.reshape(-1,1)

```

4.2 Quantity

In [30]:

```

normalizer = Normalizer()

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

normalizer.fit(X_train['quantity'].values.reshape(1,-1))

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

## reshaping
quantity_train_norm=quantity_train_norm.reshape(-1,1)
```

4.3 Number of Previously posted projects

In [31]:

```
normalizer = Normalizer()

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

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

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


## reshaping
prev_projects_train_norm=prev_projects_train_norm.reshape(-1,1)
```

Assignment 10 - Clustering

- step 1: Choose any vectorizer (data matrix) that you have worked in any of the assignments, and got the best AUC value.
- step 2: Choose any of the feature selection (https://scikit-learn.org/stable/modules/feature_selection.html)/reduction algorithms (https://scikit-learn.org/stable/modules/decomposition.html) ex: selectkbest features, pretrained word vectors, model based feature 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 (https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/) and try a different number of clusters like 2,5 etc.
    - As this is very computationally expensive, take **5k** datapoints only to perform hierarchical clustering because they do take a considerable amount of time to run.
  - **DBSCAN Clustering:**
    - Find the best 'eps' using the elbow-knee method (https://stackoverflow.com/a/48558030/4084039).
    - Take **5k** datapoints only.
- 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 in step 3.

In [32]:
```python
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039

X_tr = hstack((train_categories, train_subcategories,sklstate_train,teacher_prefix_train,
            proj_grade_train,bow_essay_train,bow_title_train,
            X_train_price_norm,prev_projects_train_norm)).tocsr()

print(X_tr.shape)


```

```
(109248, 7556)
```

In [33]:
```python
# Find 5000 features
from sklearn.feature_selection import SelectKBest
X_tr = SelectKBest(k=5000).fit_transform(X_tr,y_train)
X_tr.shape
```

Out[33]:
```
(109248, 5000)
```

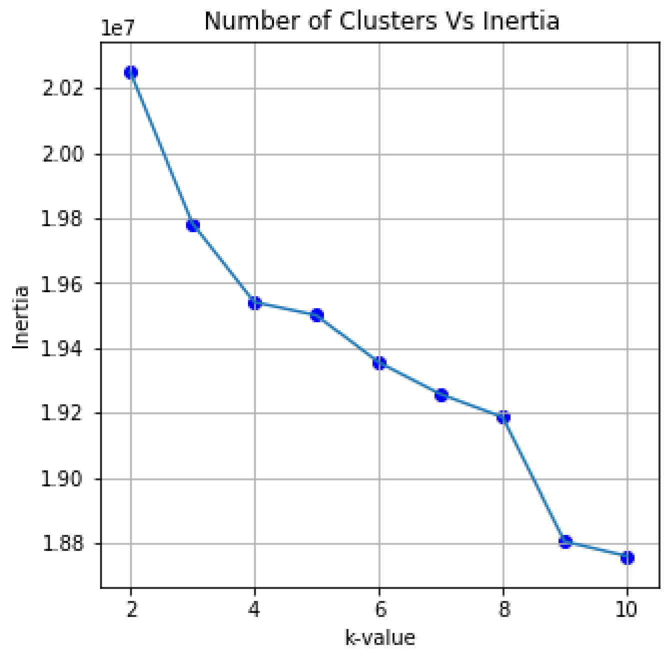
5. K-means - Clustering

In [34]:
```python
# reference : https://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html
from sklearn.cluster import MiniBatchKMeans
k=[2,3,4,5,6,7,8,9,10]
inertia=[]
for i in tqdm(k):
    kmeans=MiniBatchKMeans(n_clusters=i,init='k-means++',n_init=10,random_state=4)
    kmeans.fit(X_tr)
    inertia.append(kmeans.inertia_)

```

```
100%|██████████████████████████████████████████████████| 9/9 [00:08<00:00,  1.03it/s]
```

5.1 Plotting K vs Intertia

In [35]:
```
plt.figure(figsize=(5,5))
plt.plot(k,inertia)
plt.scatter(k,inertia,color='b')
plt.xlabel('k-value')
plt.ylabel('Inertia')
plt.title(' Number of Clusters Vs Inertia')
plt.grid()
```



Observations :

1. We see that after k=4 (elbow point) inertia decreases linearly .
2. We can take k=4 as the optimal number of cluster

In [36]:
```
# train with the best k  value
kmeans=MiniBatchKMeans(n_clusters=4,init='k-means++',n_init=10,random_state=4)
kmeans.fit(X_tr)
```

Out[36]:
```
MiniBatchKMeans(batch_size=100, compute_labels=True, init='k-means++',
        init_size=None, max_iter=100, max_no_improvement=10, n_clusters=4,
        n_init=10, random_state=4, reassignment_ratio=0.01, tol=0.0,
        verbose=0)
```

In [37]:
```
def extract_datapoints(labels,data,cl_no):
    df_=[]
    for i in range(len(labels)):
        if labels[i] ==  cl_no :
            df_.append(i)
    return data.iloc[df_]
```

In [38]:
```
cl1=extract_datapoints(kmeans.labels_,project_data,0)
cl2=extract_datapoints(kmeans.labels_,project_data,1)
cl3=extract_datapoints(kmeans.labels_,project_data,2)
cl4=extract_datapoints(kmeans.labels_,project_data,3)
```

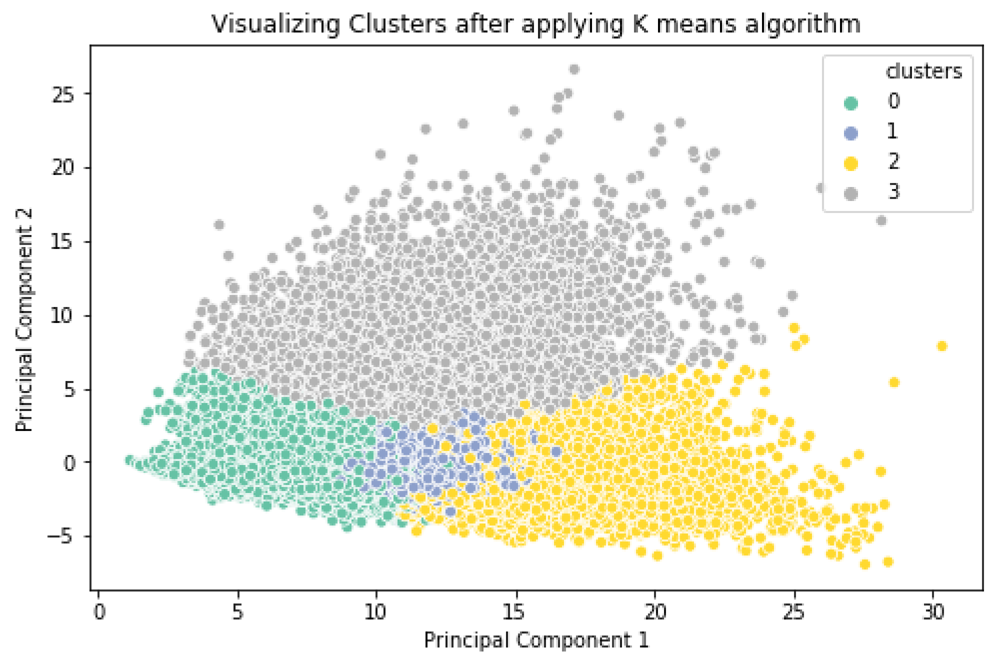
Lets visualize Clusters using TruncatedSVD

In [39]:
```
from sklearn.decomposition import TruncatedSVD
tvd = TruncatedSVD(n_components=2)
principalComponents = tvd.fit_transform(X_tr)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['Principal Component 1', 'Principal Component 2'])
```

In [40]:
```
principalDf['clusters'] = kmeans.labels_
```

In [41]:
```
plt.figure(figsize=(8,5))
sns.scatterplot(x='Principal Component 1', y='Principal Component 2', hue='clusters',
                palette="Set2",data=principalDf)
plt.title('Visualizing Clusters after applying K means algorithm')
```

Out[41]:
```
Text(0.5, 1.0, 'Visualizing Clusters after applying K means algorithm')
```



5.2 Manually summarizing each cluster

In [42]:
```
cl1.head(3)
```

Out[42]:

	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_projects	pr
0	p253737	c90749f5d961ff158d4b4d1e7dc665fc	mrs	in	2016-12-05 13:43:57	prek_2	Educational Support for English Learners at Home	My students need opportunities to practice beg...	0	
1	p258326	897464ce9ddc600bced1151f324dd63a	mr	fl	2016-10-25 09:22:10	6_8	Wanted: Projector for Hungry Learners	My students need a projector to help with view...	7	
4	p104768	be1f7507a41f8479dc06f047086a39ec	mrs	tx	2016-07-11 01:10:09	prek_2	Interactive Math Tools	My students need hands on practice in mathemat...	1	

In [43]:
```
cl2.head(3)
```

Out[43]:

	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_projects
3	p246581	f3cb9bffbba169bef1a77b243e620b60	mrs	ky	2016-10-06 21:16:17	prek_2	Techie Kindergarteners	My students need to engage in Reading and Math...	4
8	p045029	487448f5226005d08d36bdd75f095b31	mrs	sc	2016-09-25 17:00:26	prek_2	Targeting More Success in Class	My students need three devices and three manag...	28
14	p233127	424819801de22a60bba7d0f4354d0258	ms	ma	2017-02-14 16:29:10	prek_2	TABLETS CAN SHOW US THE WORLD	My students need 5 tablets for our classroom t...	15

In [44]:

```
cl3.head(3)
```

Out[44]:

	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_projects	p
2	p182444	3465aaf82da834c0582ebd0ef8040ca0	ms	az	2016-08-31 12:03:56	6_8	Soccer Equipment for AWESOME Middle School Stu...	My students need shine guards, athletic socks,...	1	
5	p154343	a50a390e8327a95b77b9e495b58b9a6e	mrs	fl	2017-04-08 22:40:43	3_5	Flexible Seating for Mrs. Jarvis' Terrific Thi...	My students need movement to be successful. Be...	1	
7	p092424	5bfd3d12fae3d2fe88684bbac570c9d2	ms	ga	2016-09-01 00:02:15	3_5	It's the 21st Century	My students need ipads to help them access a w...	7	

In [45]:

```
cl4.head(3)
```

Out[45]:

	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_projects
9	p001713	140eeac1885c820ad5592a409a3a8994	ms	nc	2016-11-17 18:18:56	prek_2	Just For the Love of Reading--\r\nPure Pleasure	My students need great books to use during Ind...	36
10	p040307	363788b51d40d978fe276bcb1f8a2b35	mrs	ca	2017-01-04 16:40:30	3_5	Reading Changes Lives	My students need books by their favorite autho...	37
20	p052326	e0c1aad1f71badeff703fadc15f57680	mrs	pa	2016-10-07 18:27:02	prek_2	Magic Carpet Ride in Our Library	My students need carpet in our library to brig...	23

5.3 Plotting Word Cloud

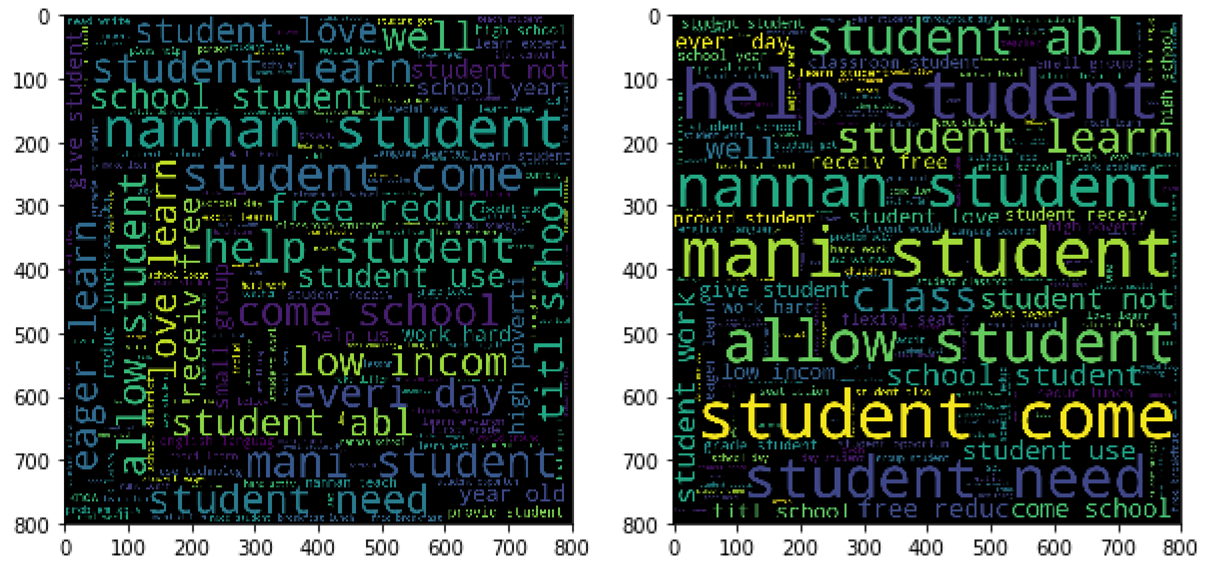
In [46]:

```python
def word_cloud(cl):
    stopwords = set(STOPWORDS)
    ##merge the text of false positive points
    word_cloud_str=' '.join(cl['processed_essay'].values)
    #create wordcloud
    wordcloud = WordCloud(width = 800, height = 800,
                background_color ='black',
                stopwords = stopwords,
                min_font_size = 10).generate(word_cloud_str)
    return wordcloud
```

Cluster 1 and Cluster 2

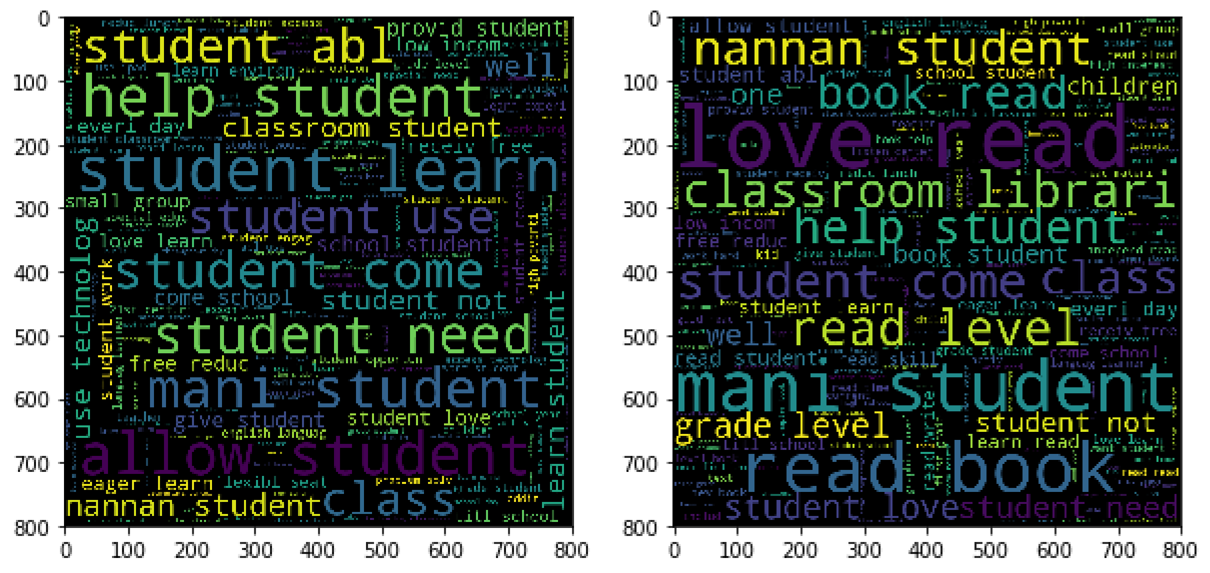
In [47]:

```python
## word cloud for cluster 1(left) and cluster2(right)
# plot the WordCloud image
# https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python
wordcloud1=word_cloud(cl1)
wordcloud2=word_cloud(cl2)
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10,5))
axes[0].imshow(wordcloud1)
axes[1].imshow(wordcloud2)
plt.show()

```



Cluster 3 and Cluster 4

In [48]:
```python
## word cloud for cluster 3(left) and cluster 4(right)
# plot the WordCloud image
# https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python
wordcloud3=word_cloud(cl3)
wordcloud4=word_cloud(cl4)
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10,5))
axes[0].imshow(wordcloud3)
axes[1].imshow(wordcloud4)
plt.show()

```



Observation :

1. we can observe difference in the word cloud of all the 4 clusters.
2. In cluster 4 we can observe words like love,read that are uncommon in other clusters.

6 Alglomerative Clustering

In [49]:
```python
### reducing our trainset to 5000 points using random sampling
randlist=random.sample(range(1,X_tr.shape[0]), 5000)
sampled_tr=X_tr[randlist].todense()
```

In [50]:
```
## converting to sparse matrix
sampled_tr=scipy.sparse.csr_matrix(sampled_tr)
```

In [51]:
```
print('Shape of dataset ', sampled_tr.shape)
```

```
Shape of dataset  (5000, 5000)
```

Observing with n_clusters=2

In [52]:
```
# reference : https://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html
from sklearn.cluster import AgglomerativeClustering

agg=AgglomerativeClustering(linkage='ward',n_clusters=2)
agg.fit(sampled_tr.toarray())

```

Out[52]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
            connectivity=None, linkage='ward', memory=None, n_clusters=2,
            pooling_func='deprecated')
```

In [53]:
```
agg.labels_
```

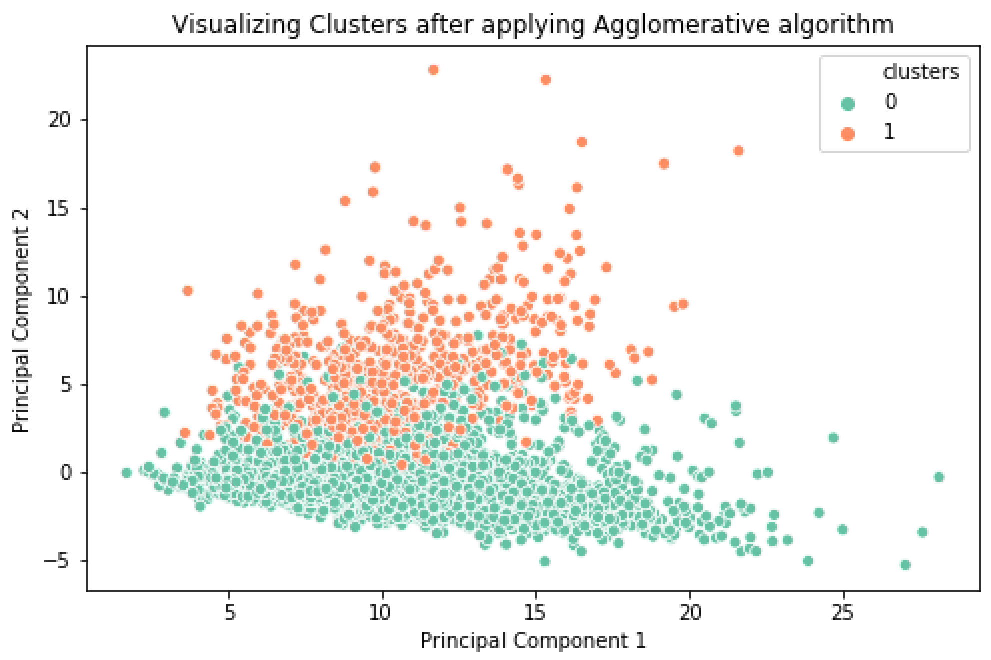
Out[53]:
```
array([0, 0, 0, ..., 1, 0, 0], dtype=int64)
```

In [54]:
```
data=project_data.iloc[randlist].reset_index()
cl1=extract_datapoints(agg.labels_,data,0)
cl2=extract_datapoints(agg.labels_,data,1)
# cl3=extract_datapoints(agg.labels_,data,2)
# cl4=extract_datapoints(agg.labels_,data,3)
# cl5=extract_datapoints(agg.labels_,data,4)
```

Lets visualize Clusters using TruncatedSVD

In [55]:
```
tvd = TruncatedSVD(n_components=2)
principalComponents = tvd.fit_transform(sampled_tr)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['Principal Component 1', 'Principal Component 2'])
principalDf['clusters'] = agg.labels_
plt.figure(figsize=(8,5))
sns.scatterplot(x='Principal Component 1', y='Principal Component 2', hue='clusters',
                palette="Set2",data=principalDf)
plt.title('Visualizing Clusters after applying Agglomerative algorithm')
```

Out[55]:
```
Text(0.5, 1.0, 'Visualizing Clusters after applying Agglomerative algorithm')
```



6.2 Manually summarizing each cluster

In [56]:
```
cl1.head(3)
```

Out[56]:

	index	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_proj
0	107187	p002957	4e31fea7fb07f1a24e389286bbd16b73	mrs	hi	2016-07-02 20:00:28	prek_2	Blooming Second Graders Need STEM!\r\n	My students need i-pad mini tablets to explore...	
1	91432	p213667	4149cae86b4eec56b5afa6cc25249dbb	mrs	co	2017-03-01 13:09:53	6_8	Entice our SENSES!	My students need a variety of sensory and game...	
2	68006	p128372	9e536466e3d1f5d9ffcd3be4fee42d1d	mrs	nc	2016-07-08 10:21:25	3_5	Sitting, Standing, and Bouncing Our Way to Suc...	My students need additional options for altern...	

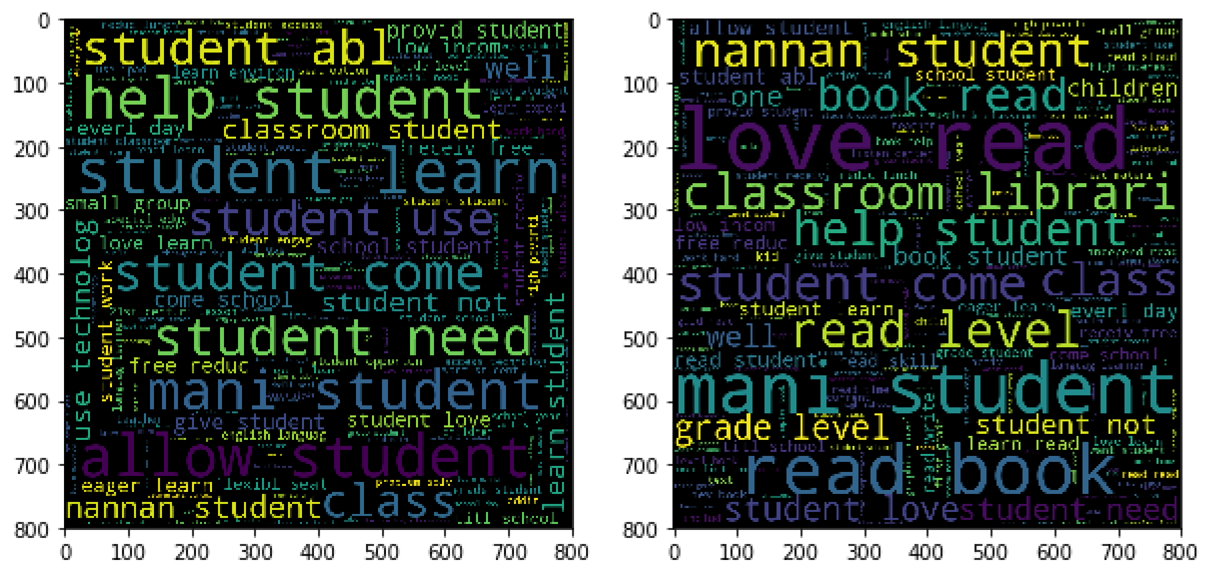
In [57]:
```
cl2.head(3)
```

Out[57]:

	index	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_p
12	109093	p013168	0a3d369d37b820e300d5dca46ad62edb	ms	ny	2017-04-05 19:00:55	6_8	Sustain in the Summer	My students need books, snacks, and cinch sack...	
15	66527	p254646	891607ee651ce7db8b3946de6d560281	mrs	or	2016-09-01 00:01:27	prek_2	iPads For Access to Valuable Math and Reading ...	My students need iPads with cases and headphon...	
25	3776	p247608	0034e3cf3ea440efa7e85ea6aa5b867a	ms	tx	2016-08-02 17:50:02	prek_2	My Growing Kinder Garden	My students need a variety of different levele...	

6.3 Plotting Word Cloud

In [58]:
```python
## word cloud for cluster 1(left) and cluster2(right)
# plot the WordCloud image
# https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python
wordcloud1=word_cloud(cl1)
wordcloud2=word_cloud(cl2)
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10,5))
axes[0].imshow(wordcloud3)
axes[1].imshow(wordcloud4)
plt.show()

```



Observations :

1.The clusters have some common words like student,help.
2.We observe words like love,read,level,classroom,library in cluster 2 which is not in cluster 1.

Observing with n_clusters=5

In [59]:
```python
# reference : https://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html
agg=AgglomerativeClustering(linkage='ward',n_clusters=5)
agg.fit(sampled_tr.toarray())

```

Out[59]:
```
AgglomerativeClustering(affinity='euclidean', compute_full_tree='auto',
            connectivity=None, linkage='ward', memory=None, n_clusters=5,
            pooling_func='deprecated')
```

In [60]:
```
agg.labels_
```

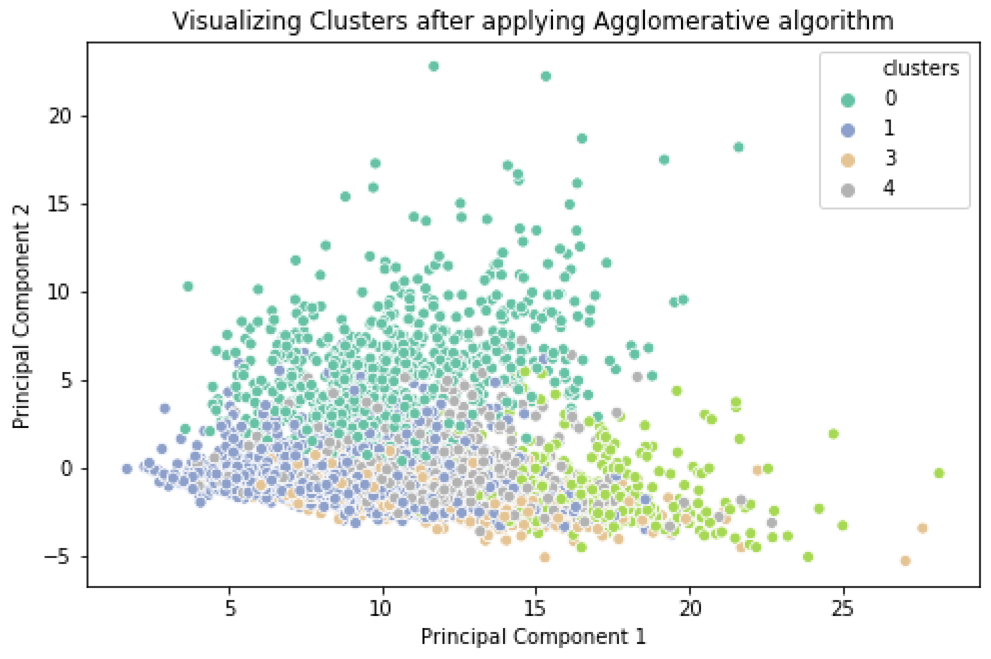
Out[60]:
```
array([4, 4, 3, ..., 0, 4, 1], dtype=int64)
```

In [61]:
```
data=project_data.iloc[randlist].reset_index()
cl1=extract_datapoints(agg.labels_,data,0)
cl2=extract_datapoints(agg.labels_,data,1)
cl3=extract_datapoints(agg.labels_,data,2)
cl4=extract_datapoints(agg.labels_,data,3)
cl5=extract_datapoints(agg.labels_,data,4)
```

Lets visualize Clusters using TruncatedSVD

In [62]:
```
tvd = TruncatedSVD(n_components=2)
principalComponents = tvd.fit_transform(sampled_tr)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['Principal Component 1', 'Principal Component 2'])
principalDf['clusters'] = agg.labels_
plt.figure(figsize=(8,5))
sns.scatterplot(x='Principal Component 1', y='Principal Component 2', hue='clusters',
                palette="Set2",data=principalDf)
plt.title('Visualizing Clusters after applying Agglomerative algorithm')
```

Out[62]:
```
Text(0.5, 1.0, 'Visualizing Clusters after applying Agglomerative algorithm')
```



6.2 Manually summarizing each cluster

In [63]:
```
cl1.head(3)
```

Out[63]:

	index	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_p
12	109093	p013168	0a3d369d37b820e300d5dca46ad62edb	ms	ny	2017-04-05 19:00:55	6_8	Sustain in the Summer	My students need books, snacks, and cinch sack...	
15	66527	p254646	891607ee651ce7db8b3946de6d560281	mrs	or	2016-09-01 00:01:27	prek_2	iPads For Access to Valuable Math and Reading ...	My students need iPads with cases and headphon...	
25	3776	p247608	0034e3cf3ea440efa7e85ea6aa5b867a	ms	tx	2016-08-02 17:50:02	prek_2	My Growing Kinder Garden	My students need a variety of different levele...	

In [64]:
```
cl2.head(3)
```

Out[64]:

	index	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_pro
3	61965	p091105	8751e989c023a57640c4de7854c73b67	ms	ga	2017-01-18 11:58:37	6_8	Technology for Daily Differentiation in the Cl...	My students need four Acer Chromebooks to use ...	
4	13165	p031997	825556ed7f83ee150bd6ab286ecc3b54	mr	tx	2016-07-20 15:42:06	3_5	It's Time to Read	My students need books so they can read at hom...	
7	7982	p170411	21c7d2f693447d32d574ad09a96f466e	ms	va	2017-01-07 16:10:08	prek_2	Rock PAPER Scissors!	My students need paper so they can write about...	

In [65]:
```
cl3.head(3)
```

Out[65]:

	index	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_postec
8	84661	p155575	31a5deacca7253007792b5825a188a30	ms	wa	2016-09-05 17:51:04	9_12	Life Skills Bistro!	My students need hands on experiences in the c...	
16	42206	p069761	4e9a1e02c945c77042430ad2dc3e7ae1	ms	nm	2016-10-20 19:41:05	3_5	In \"Time\" We Can Grow in Our Reading and Math	My students need a class set of timers to impr...	

In [66]:
```
cl5.head(3)
```

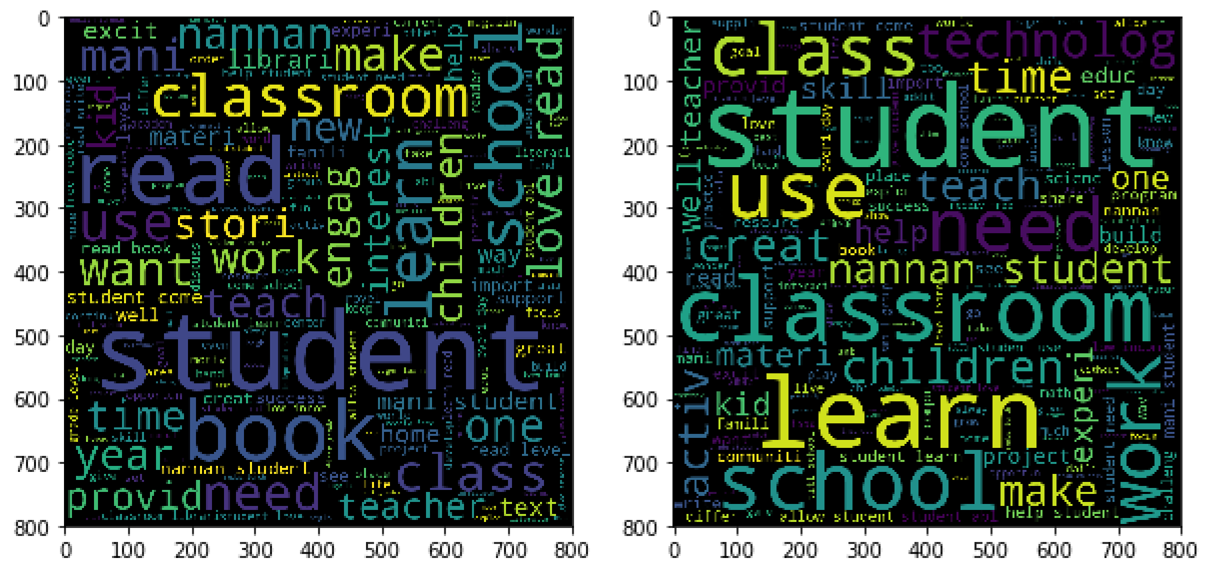
Out[66]:

	index	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_pro
0	107187	p002957	4e31fea7fb07f1a24e389286bbd16b73	mrs	hi	2016-07-02 20:00:28	prek_2	Blooming Second Graders Need STEM!\r\n	My students need i-pad mini tablets to explore...	
1	91432	p213667	4149cae86b4eec56b5afa6cc25249dbb	mrs	co	2017-03-01 13:09:53	6_8	Entice our SENSES!	My students need a variety of sensory and game...	
11	98924	p106031	2c8a2479cc536695c604ee088cfcdd1a	ms	pa	2017-01-05 09:52:00	prek_2	ALEXA! ALEXA! May We Ask a Question? Please!	My students need prizes and an Amazon Echo to ...	

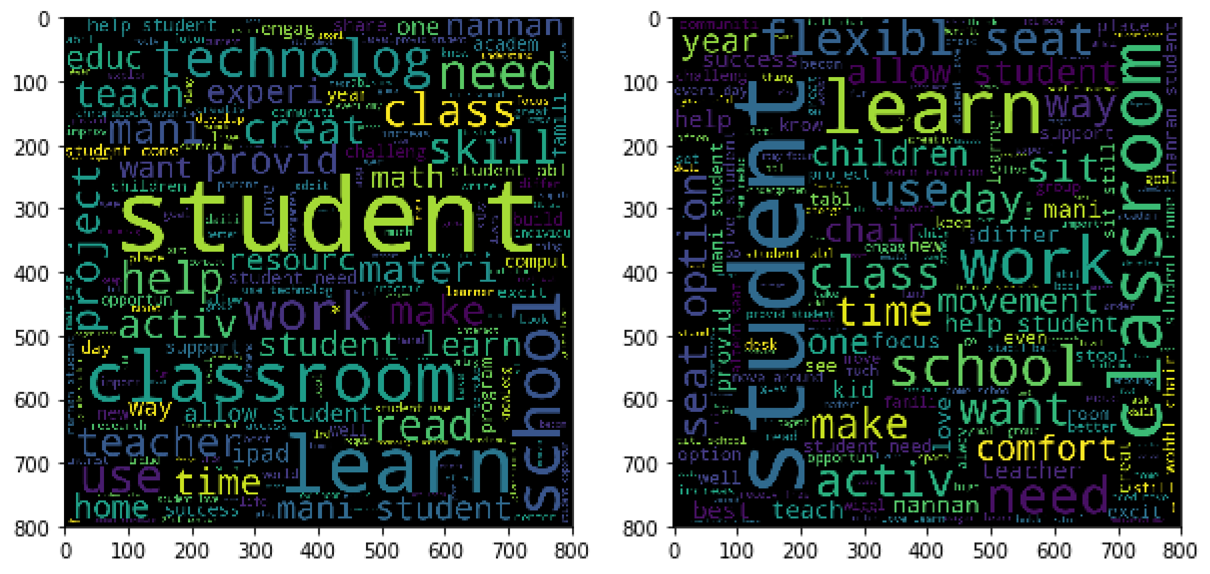
6.3 Plotting Word Cloud

In [67]:

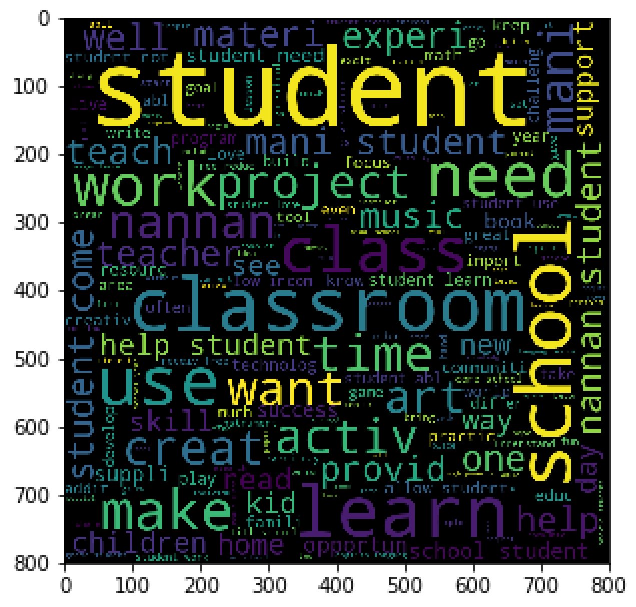
```
## word cloud for cluster 1(left) and cluster2(right)
# plot the WordCloud image
# https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python
wordcloud1=word_cloud(cl1)
wordcloud2=word_cloud(cl2)
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10,5))
axes[0].imshow(wordcloud1)
axes[1].imshow(wordcloud2)
plt.show()

```



In [68]:
```python
## word cloud for cluster 3(left) and cluster4(right)
# plot the WordCloud image
# https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python
wordcloud3=word_cloud(cl3)
wordcloud4=word_cloud(cl4)
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(10,5))
axes[0].imshow(wordcloud3)
axes[1].imshow(wordcloud4)
plt.show()
```



In [69]:
```python
## word cloud for cluster 1(left) and cluster2(right)
# plot the WordCloud image
# https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python
wordcloud5=word_cloud(cl5)
plt.figure(figsize = (10,5), facecolor ="none")
plt.imshow(wordcloud5)
plt.show()
```



Observations :

```
1. Using number of clusters 2 is better than using 5 as we dont see much separation between points.
2. Observing from word clouds we see all the 5 clusters have most used words as student and classroom,
   the uncommon words are read,technology,school,time etc
```

7.DBSCAN

Finding the best ep using Elbow method :

```
Steps:
        1.Since our data contains only 5k points let our min point be ln(5000)
        (i.e) 8.5 .We can keep our min point to be 10
        2.Find the index of 10th nearest neighbour of each point and the distance (kdistance)
        3.plot the index vs the kth distance
```

In [70]:

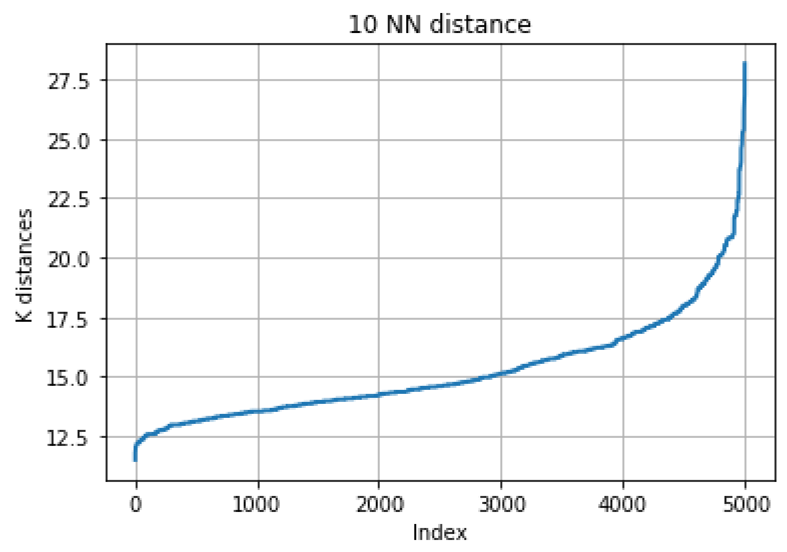
```python
from sklearn.neighbors import NearestNeighbors
_10nn=NearestNeighbors(n_neighbors=10).fit(sampled_tr)
dist=_10nn.kneighbors(sampled_tr)
```

In [71]:

```python
points_=[i[9] for i in dist[1]]
distance_10=[i[9] for i in dist[0]]
```

In [72]:

```python
points_.sort()
distance_10.sort()
#https://stackoverflow.com/questions/12893492/choosing-eps-and-minpts-for-dbscan-r
# plotting sorted distances against points to select eps
plt.plot(points_,distance_10,linewidth=2)
plt.xlabel('Index')
plt.ylabel('K distances')
plt.title('10 NN distance')
plt.grid()
```



Observation :

```
1.Using elbow method we see that the optimal distance is at 20.
```

In [73]:
```
## Train DBSCAN
from sklearn.cluster import DBSCAN
dbscan=DBSCAN(eps=20,min_samples=10,n_jobs=-1)
dbscan.fit(sampled_tr)
```

Out[73]:
```
DBSCAN(algorithm='auto', eps=20, leaf_size=30, metric='euclidean',
    metric_params=None, min_samples=10, n_jobs=-1, p=None)
```

7.2 Manually summarizing each cluster

In [74]:
```
x=[]
for i in dbscan.labels_:
    if i not in x:
        x.append(i)
```

In [75]:
```
print(x)
print('Total number of clusters is 1 and few points are labelled noise ')
print()
```

```
[0, -1]
Total number of clusters is 1 and few points are labelled noise
```

In [76]:
```
data=project_data.iloc[randlist].reset_index()
cl1=extract_datapoints(dbscan.labels_,data,0)
cl_outlier=extract_datapoints(dbscan.labels_,data,-1)
```

In [77]:
```
## few points in cluster 1
cl1.head(3)
```

Out[77]:

	index	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_posted_proj
0	107187	p002957	4e31fea7fb07f1a24e389286bbd16b73	mrs	hi	2016-07-02 20:00:28	prek_2	Blooming Second Graders Need STEM!\r\n	My students need i-pad mini tablets to explore...	
1	91432	p213667	4149cae86b4eec56b5afa6cc25249dbb	mrs	co	2017-03-01 13:09:53	6_8	Entice our SENSES!	My students need a variety of sensory and game...	
2	68006	p128372	9e536466e3d1f5d9ffcd3be4fee42d1d	mrs	nc	2016-07-08 10:21:25	3_5	Sitting, Standing, and Bouncing Our Way to Suc...	My students need additional options for altern...	

In [78]:

```
## outlier points
cl_outlier.head(3)
```

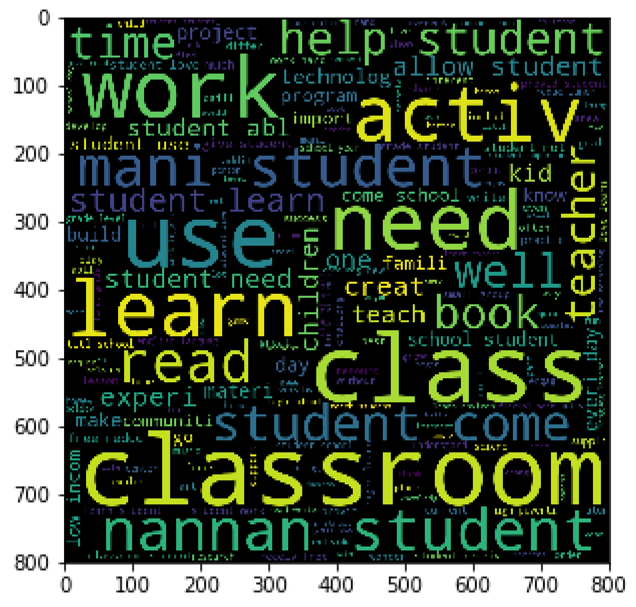
Out[78]:

	index	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_category	project_title	project_resource_summary	teacher_number_of_previously_poste
8	84661	p155575	31a5deacca7253007792b5825a188a30	ms	wa	2016-09-05 17:51:04	9_12	Life Skills Bistro!	My students need hands on experiences in the c...	
200	60376	p140488	b18f80571ffe7a2cb90d1ae525c937f2	mrs	ca	2016-10-03 17:08:25	6_8	Central \"Storage\" Unit For All My Students a...	My students need a classroom computer which ca...	
221	8945	p130107	a349be46eb5368fac9c540380940d89b	ms	wa	2016-10-02 18:38:08	3_5	Discovering Math Through Art	My students need art supplies to create projec...	

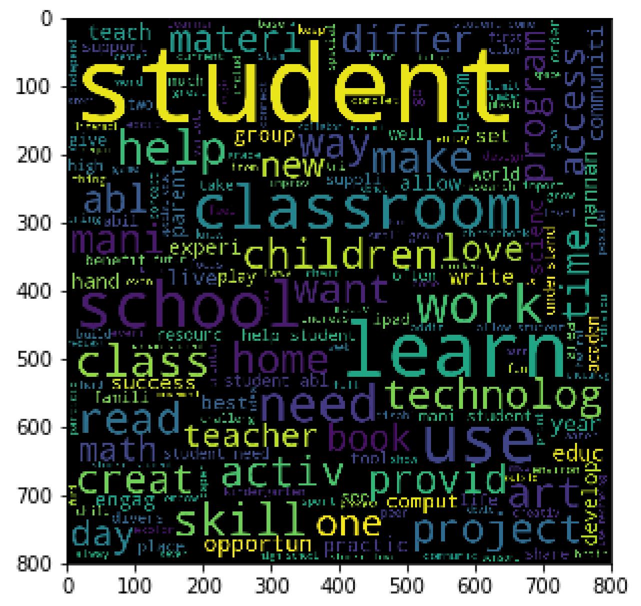
7.3 Word cloud for cluster 1

In [79]:

```
## word cloud for cluster 1(left)
# plot the WordCloud image
# https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python
wordcloud1=word_cloud(cl1)
plt.figure(figsize = (10,5), facecolor ="none")
plt.imshow(wordcloud1)
plt.show()
```



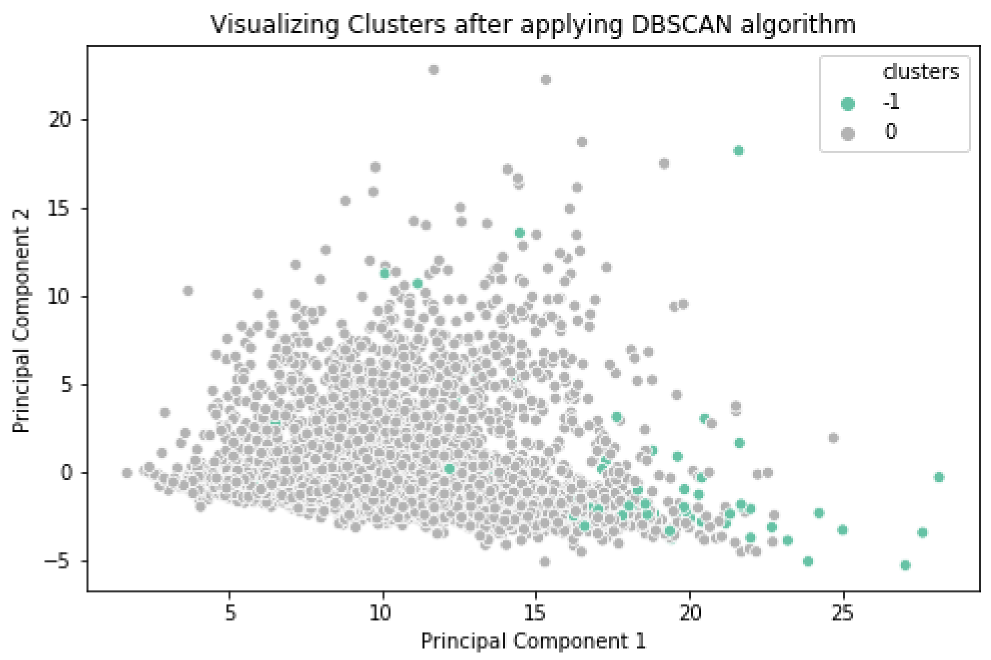
In [80]:
```
## word cloud for Noise points
# plot the WordCloud image
# https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python
outliercloud=word_cloud(cl_outlier)
plt.figure(figsize = (10,5), facecolor ="none")
plt.imshow(outliercloud)
plt.show()
```



Lets visualize Clusters using TruncatedSVD

In [81]:
```
tvd = TruncatedSVD(n_components=2)
principalComponents = tvd.fit_transform(sampled_tr)
principalDf = pd.DataFrame(data = principalComponents
             , columns = ['Principal Component 1', 'Principal Component 2'])
principalDf['clusters'] = dbscan.labels_
plt.figure(figsize=(8,5))
sns.scatterplot(x='Principal Component 1', y='Principal Component 2', hue='clusters',
                palette="Set2",data=principalDf)
plt.title('Visualizing Clusters after applying DBSCAN algorithm')
```

Out[81]:
```
Text(0.5, 1.0, 'Visualizing Clusters after applying DBSCAN algorithm')
```



Conclusions :

Kmeans :

1. Using elbow method we found the best number of cluster to be 4
2. After reducing the dimensions using truncatedSVD we see that intercluster distances is very small between clusters.
3. Found differences in the words in essay between clusters

Agglomorative :

1. Due to memory constraints number of datapoint used is 5k.
2. Number of clusters as 2 separates points better than clusters 5 .
3. Found differences in the words in essay between clusters

DBSCAN:

1. Due to memory constraints number of datapoint used is 5k.
2. Using elbow knee method we found best eps to be 20 and considering min points as 10 by default.
3. We observed that DBSCAN clusters 96% points into 1 cluster and rest 4% as noisy points

Since the clusters are having very small intercluster distance ideally optimal number of clusters is 1 as observed in DBSCAN.

https://stackoverflow.com/questions/43784903/scikit-k-means-clustering-performance-measure (https://stackoverflow.com/questions/43784903/scikit-k-means-clustering-performance-measure)

https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python (https://stackoverflow.com/questions/42818361/how-to-make-two-plots-side-by-side-using-python)

https://stackoverflow.com/questions/12893492/choosing-eps-and-minpts-for-dbscan-r (https://stackoverflow.com/questions/12893492/choosing-eps-and-minpts-for-dbscan-r)

https://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html (https://scikit-learn.org/stable/auto_examples/cluster/plot_mini_batch_kmeans.html)

https://towardsdatascience.com/cluster-analysis-create-visualize-and-interpret-customer-segments-474e55d00ebb (https://towardsdatascience.com/cluster-analysis-create-visualize-and-interpret-customer-segments-474e55d00ebb)

In [ ]:

```
```