DonorsChoose

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

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, 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). **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories for the project. **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. **Example:** • `My students need hands on literacy materials to manage sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*

Feature	Description
`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]:

```
%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

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

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

1.1 Reading Data

In [2]:

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

In [3]:

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

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

In [4]:

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

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

Out[4]:

	id	description	quantity	price
		LC652 - Lakeshore Double Space Mobile Drying		

	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

In [5]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: 
https://stackoverflow.com/a/47301924/4084039

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

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`

In [6]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: 
https://stackoverflow.com/a/47301924/4084039

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

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "Care & H
unger"]
        if 'The' in j.split(): # this will split each of the catogory based on space "Math & Scienc
e"=> "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('&','_')
    sub_cat_list.append(temp.strip())

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

In [7]:

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

In [8]:

```
project_data.head(2)
```

Out[8]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Grades P
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Grade

In [9]:

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

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/reduction algorithms 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 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.
    ● 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.

2. Clustering

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

2.1 Splitting data into Train and cross validation(or test): Stratified Sampling

In [10]:

```
sample_data=project_data.sample(10000)
sample_data.shape
from sklearn.model_selection import train_test_split
data=sample_data

y_train=data.project_is_approved
x_train=data.drop('project_is_approved',axis=1)



print("shape of train data ")
print(x_train.shape)
print(y_train.shape)
```

```
shape of train data
(10000, 19)
(10000,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [11]:

```
def veccat(x):
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.preprocessing import Normalizer
    from scipy.sparse import hstack
    vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=T
rue)
    vectorizer.fit(x['clean_categories'].values)
    print(vectorizer.get_feature_names())



    categories_one_hot = vectorizer.transform(x['clean_categories'].values)
    print("Shape of matrix after one hot encodig ",categories_one_hot.shape)

    vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
binary=True)
    vectorizer.fit(x['clean_subcategories'].values)
    print(vectorizer.get_feature_names())



    subcategories_one_hot = vectorizer.transform(x['clean_subcategories'].values)
    print("Shape of matrix after one hot encodig ",subcategories_one_hot.shape)


    vectorizer = CountVectorizer(lowercase=False, binary=True)
    vectorizer.fit(x['school_state'].values)
    print(vectorizer.get_feature_names())



    state_one_hot = vectorizer.transform(x['school_state'].values)
    print("Shape of matrix after one hot encodig ",state_one_hot.shape)

    x = x.replace(np.nan, '', regex=True)
    vectorizer = CountVectorizer( vocabulary=['Mrs.', 'Ms.','Mr.','Teacher','Dr.'],lowercase=False)
    vectorizer.fit(x['teacher_prefix'].values)
    print(vectorizer.get_feature_names())
    prefix_one_hot = vectorizer.transform(x['teacher_prefix'].values)
    print("Shape of matrix after one hot encodig ",prefix_one_hot.shape)

    vocab=[]
    for i in x['project_grade_category'].values:
        vocab.append(i)
    v_set=set(vocab)
    vocab=list(v_set)
    vectorizer = CountVectorizer(vocabulary=vocab,lowercase=False)
    vectorizer.fit(x['project_grade_category'].values)




    grade_one_hot = vectorizer.transform(x['project_grade_category'].values)
    print("Shape of matrix after one hot encodig ",grade_one_hot.shape)

    price_scalar =  Normalizer(copy=False,norm='l2')
    price_scalar.fit(x['price'].values.reshape(1,-1)) # finding the mean and standard deviation of
this data



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

    projects_scalar = Normalizer(copy=False,norm='l2')
    projects_scalar.fit(x['teacher_number_of_previously_posted_projects'].values.reshape(1,-1)) # f
inding the mean and standard deviation of this data

    # Now standardize the data with above maen and variance.
    projects_standardized =
projects_scalar.transform(x['teacher_number_of_previously_posted_projects'].values.reshape(1, -1))
    projects_standardized =np.transpose(projects_standardized)

    qty_scalar= Normalizer(copy=False,norm='l2')
    qty_scalar.fit(x['quantity'].values.reshape(1,-1)) # finding the mean and standard deviation of
this data


    # Now standardize the data with above maen and variance.
    qty_standardized = qty_scalar.transform(x['quantity'].values.reshape(1, -1))
    qty_standardized=np.transpose(qty_standardized)

    X1 = hstack((categories_one_hot, subcategories_one_hot,state_one_hot,prefix_one_hot,grade_one_h
ot, price_standardized,projects_standardized,qty_standardized))
    print(X1.shape)
    return(X1)
```

In [12]:

```
x=veccat(x_train)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (10000, 9)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (10000, 30)
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', '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 after one hot encodig  (10000, 51)
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (10000, 5)
Shape of matrix after one hot encodig  (10000, 4)
(10000, 102)
```

```
[illegible]
```

In [13]:

```
def features(x):
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.preprocessing import MinMaxScaler
    from scipy.sparse import hstack
    vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=T
rue)
    vectorizer.fit(x['clean_categories'].values)

    feature_list.extend(vectorizer.get_feature_names())
    vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
binary=True)
    vectorizer.fit(x['clean_subcategories'].values)

    feature_list.extend(vectorizer.get_feature_names())
    vectorizer = CountVectorizer(lowercase=False, binary=True)
    vectorizer.fit(x['school_state'].values)

    feature_list.extend(vectorizer.get_feature_names())
    x = x.replace(np.nan, '', regex=True)
    vectorizer = CountVectorizer( vocabulary=['Mrs.', 'Ms.','Mr.','Teacher','Dr.'],lowercase=False)
    vectorizer.fit(x['teacher_prefix'].values)

    feature_list.extend(vectorizer.get_feature_names())
    vocab=[]
    for i in x['project_grade_category'].values:
        vocab.append(i)
    v_set=set(vocab)
    vocab=list(v_set)
    vectorizer = CountVectorizer(vocabulary=vocab,lowercase=False)
    vectorizer.fit(x['project_grade_category'].values)
    feature_list.extend(vectorizer.get_feature_names())
    feature_list.append("price")
    feature_list.append("teacher_number_of_previously_posted_projects")
    feature_list.append("quantity")
```

In [14]:

```
feature_list=[]
features(x_train)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [15]:

```
import re

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

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

In [16]:

```
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', 'e
ach', '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', "do
esn'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"]
```

In [17]:

```python
def preprocessing(x):
    import nltk
    nltk.download('stopwords')
    from tqdm import tqdm
    preprocessed_essays = []
    # tqdm is for printing the status bar
    for sentence in tqdm(x.values):
        sent = decontracted(sentence)
        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.lower() for e in sent.split() if e.lower() not in stopwords)
        preprocessed_essays.append(sent.strip())
    return preprocessed_essays
```

In [18]:

```python
train_essay=[]

train_title=[]

train_essay=preprocessing(x_train['essay'])


train_title=preprocessing(x_train['project_title'])
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|██████████████████████████████████████████████████████████████████████████| 10000/10000
[00:09<00:00, 1093.20it/s]
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|█████████████████████████████████████████████████████████████████████████| 10000/10000
[00:00<00:00, 26172.65it/s]
```

Encoding using BOW

In [19]:

```python
from scipy.sparse import hstack
from sklearn.preprocessing import StandardScaler
```

```
def bow_text(train_essay,train_title,x1):
    from scipy.sparse import hstack

    vectorizer = CountVectorizer(ngram_range=(1,2),min_df=10,max_features=5000)

    vectorizer.fit(train_essay)

    text_bow=vectorizer.transform(train_essay)



    vectorizer= CountVectorizer()
    vectorizer.fit(train_title)
    title_bow=vectorizer.transform(train_title)



    x1 = hstack((x1,text_bow,title_bow )).tocsr()




    return x1
```

In [20]:

```
# Data matrix using BOW
x1=bow_text(train_essay,train_title,x)
print(x1.shape)
```

```
(10000, 10329)
```

2.4 Dimensionality Reduction on the selected features

In [21]:

```
# 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, f_classif
x_train = SelectKBest(f_classif, k=5000).fit_transform(x1, y_train)
x_train.shape
```

```
C:\Users\ADMIN\Anaconda3\lib\site-packages\sklearn\feature_selection\univariate_selection.py:114:
UserWarning:

Features [0 0 0 0 0 0 0 0] are constant.
```

Out[21]:

```
(10000, 5000)
```

In [22]:

```
import wordcloud
from wordcloud import WordCloud
import matplotlib.pyplot as plt
%matplotlib inline
```

In [23]:

```
def cloud(centroids):
    for i in range(0, len(centroids)):
        wordcloud = WordCloud(background_color = 'white')
        kMeansWordCloud = wordcloud.generate_from_frequencies(tuple(tuple(centroids.T[i].reset_inde
x().values)))
        plt.figure()
        plt.imshow(kMeansWordCloud)
        plt.axis("off")
```

2.5 Apply Kmeans

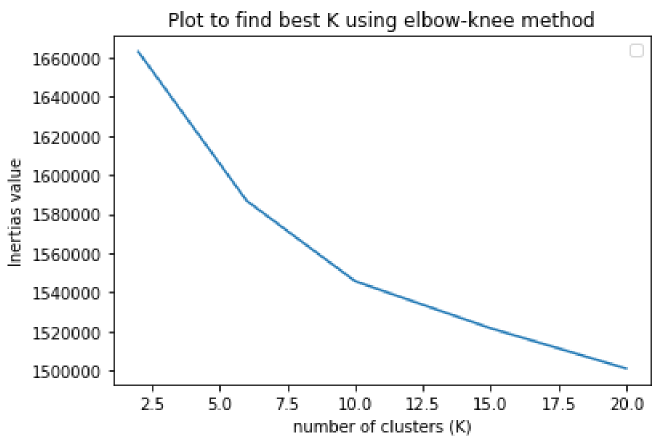
In [24]:

```
# 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 KMeans
def getLosses(K, data_matrix):
    inertias = []

    for k in K:
    #Building and fitting the model
        kmeanModel = KMeans(n_clusters=k).fit(data_matrix)
        inertias.append(kmeanModel.inertia_)
    return inertias
def plotGraph(K, inertias):
    plt.plot(K, inertias)

    plt.title("Plot to find best K using elbow-knee method")
    plt.xlabel('number of clusters (K)')
    plt.ylabel('Inertias value')
    plt.legend()
```

In [25]:

```
K = [2,6,10,15,20]
losses = getLosses(K, x_train)
plotGraph(K, losses)
```

```
No handles with labels found to put in legend.
```



The optimal k using elbow method is 6

In [26]:

```
kmeans = KMeans(n_clusters=6, random_state=0, n_jobs=-1).fit(x_train)
centroids = pd.DataFrame(kmeans.cluster_centers_)
centroids = pd.DataFrame(kmeans.cluster_centers_)
```

In [27]:

```
# Manually getting datapints from each cluster
#Code Reference : https://github.com/dileepteja3/Clustering-on-Donors-
choose/blob/master/dileep.teja3%40gmail.com_10.pdf

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

In [28]:

```
print('%s\n'%(cluster1[0]))
print('%s\n'%(cluster2[0]))
print('%s\n'%(cluster3[0]))
print('%s\n'%(cluster4[0]))
print('%s\n'%(cluster5[0]))
print('%s\n'%(cluster6[0]))
```

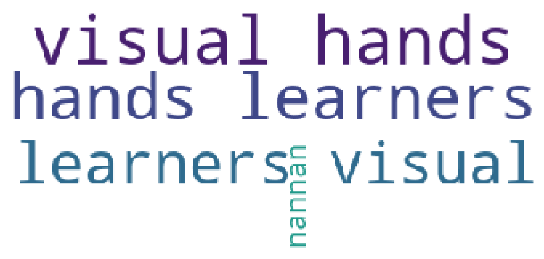
```
visual hands learners visual hands learners visual hands learners visual hands learners visual han
ds learners visual hands learners visual hands learners visual hands learners visual hands learner
s visual hands learners visual hands learners visual hands learners visual hands learners visual h
ands learners visual hands learners visual hands learners visual hands learners visual hands learn
ers visual hands learners visual hands learners visual hands learners visual hands learners visual
hands learners visual hands learners visual hands learners visual hands learners visual hands lear
ners visual hands learners visual hands learners visual hands learners visual hands learners visua
l hands learners visual hands learners visual hands learners visual hands learners visual hands le
arners visual hands learners visual hands learners visual hands learners visual hands learners vis
ual hands learners visual hands learners visual hands learners visual hands learners visual hands 
learners visual hands learners visual hands learners visual hands learners visual hands learners v
isual hands learners visual hands learners visual hands learners visual hands learners visual hand
s learners visual hands learners visual hands learners visual hands learners visual hands learners
visual hands learners visual hands learners visual hands learners visual hands learners visual han
ds learners visual hands learners visual hands learners nannan

students eager learn also active social teach title 1 school rural arizona 75 students receive 
free reduced lunch high ell population lot students never home town coming michigan hard believe n
ever seen snow student unique learning style want make sure students accommodated classroom year t
eaching math two classes second graders switch half way day two teachers mentioned earlier group d
ifferent active love move education constantly changing students needs becoming diverse sitting de
sk day not cutting lot students anymore year implemented flexible seating classroom allows student
s make choice working typically students come together carpet lesson able move working spot feel w
ould best allow work far little rugs little cushions towels students sit students also choose sit 
tradition table chairs seen great results new classroom set excited keep growing classroom 
everything purchased flexible seating pocket much would love able get students everything need sim
ply not realistic items project allow students physical outlet learning working times able wiggle 
bounce without distracting around truly believe materials help students better learners allow meet
needs students nannan

teach mostly 9th grade students first year high school exciting confusing challenging hopefully fu
n different scenario day class try make earth science relevant possible high school diverse mixtur
e cultures races economic backgrounds ranging abject poverty comfortable 20 students receive free 
reduced meals school district largely blue collar neighborhood people pull together others times n
eed remember going parents buy school supplies remember excitement kid taking everything bag organ
izing imagine come home environment resources not available able fund project benefit many ways st
udent not necessary materials class sets negative learning experience knowing come class beautiful
school supplies including notebook paper pen calculator makes student confident willing learn
```

```
addition many students suffer learning challenges adhd two balance balls intended help students ma
intain focus class numerous studies demonstrated student attention subsequent success improve
sitting balls excited see results nannan

room 18 wonderfully diverse classroom full eager learners one greatest things us friends world ame
rica europe asia learn together play together grow together look forward centers love meeting teac
her reading groups welcome homework even ask extra homework take step classroom see learning happe
ning every space quickly becoming independent learners responsible learning others great focus com
municating one another problem solving one thing learning differences make us accept not students
enjoy things understand not students learn way foundation makes us special classroom ready move 21
st century accessibility nabi allow students extend learning using academic apps math science read
ing also help students need extra stimulation electronics help focus headphones help drown sound l
earning taking place students focus work able use variety technology real world skill benefit
students need think future beyond classroom students using tablets laptops computers technology ga
dgets learn home time transition classroom space students succeed excel nannan

remember first time earned trophy ribbon not athlete imagine could reading team get rewarded showi
ng knowledge entire school district students read books join reading team compete reading battle s
tudents shining stars need little polish serve title 1 school 85 students live poverty 40 speak en
glish second language biggest obstacle access books parents cannot afford buy books often going pu
blic library involves adult able willing take school library refuge place get many materials need
free students rise challenge give sunshine state books read give reading charm collect charms enco
urage read 15 titles students get incentives along way reward parties reach goals enough copies bo
oks keep engaged difficultly offer battle book club student grades 3 5 wishes join club free read
books make detailed questions like jeopardy game students compete teams 5 answer questions correct
ly fun make friends become reading athletes school make learning look cool end school year take te
am five students club compete district wide battle books competition students chance put skills te
st incredibly bright students work team battle trophy biggest things learn sportsmanship teamwork
following rules fun important study hard proud academic achievement donation project allow offer m
ultiple copies title sunshine state list students keep interested reading learning directly impact
ing 30 students indirectly 350 core group reads books encourages others read books future
generations benefit

teach 6th grade students come various backgrounds school comprised mostly students receive free re
duced price lunch students not basic personal items work team provide students sweet respectful st
udents need shown love role models school attend elementary junior high school made k 8th grade st
udents students struggle getting supplies everyday use struggle provide manipulatives teaching con
cepts due limited budget dictionary place success comes work hard work price must pay success
think accomplish anything willing pay price said vince lombard five dictionaries classroom 12
years old dilapidated beyond repair students need new dictionaries help spelling writing dictionar
y set would allow students learn use reference books properly allow edit papers defines success ha
rd work right resources students right resources sure work hard meet expectations successful
writers nannan
```

Wordcloud

In [29]:

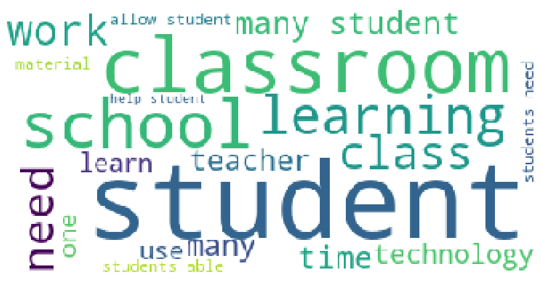
```python
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

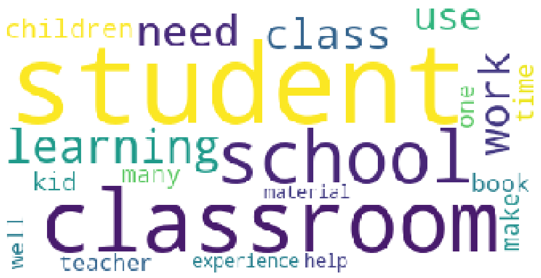


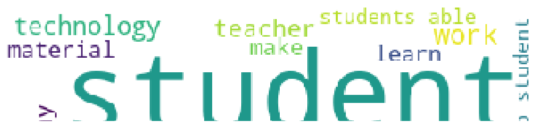
In [30]:

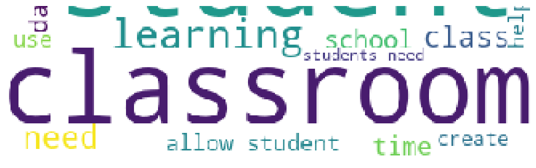
```python
words=''
```

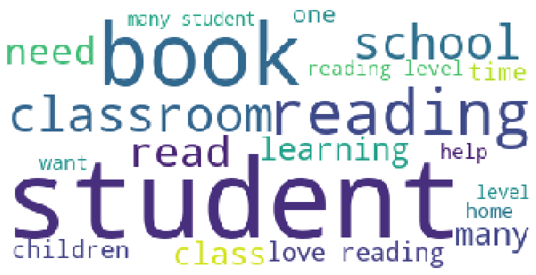
```
words=
for i in cluster2:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()

words=''
for i in cluster3:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()

words=''
for i in cluster4:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()

words=''
for i in cluster5:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
for i in cluster6:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

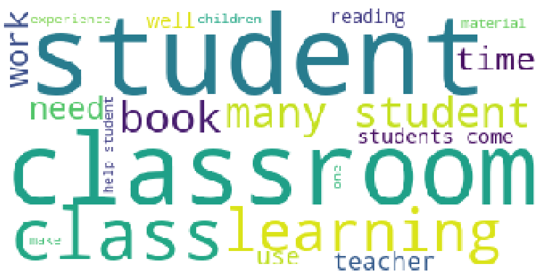












2.6 Apply AgglomerativeClustering

In [31]:

```
# 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


# K=2
x_new_train=x_train[:5000]
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
cluster.fit_predict(x_new_train.toarray())
```

Out[31]:

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

In [32]:

```
# Manually printing the data points
cluster1 = []
cluster2 = []

for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(train_essay[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(train_essay[i])
```

In [33]:

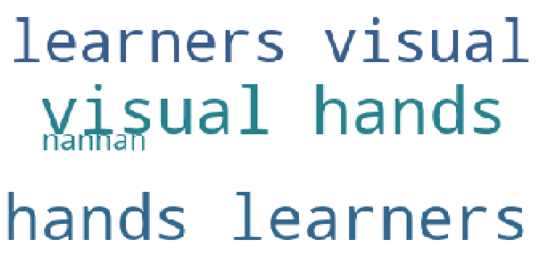
```
print('%s\n'%(cluster1[0]))
print('%s\n'%(cluster2[0]))
```

```
visual hands learners visual hands learners visual hands learners visual hands learners visual han
ds learners visual hands learners visual hands learners visual hands learners visual hands learner
s visual hands learners visual hands learners visual hands learners visual hands learners visual h
ands learners visual hands learners visual hands learners visual hands learners visual hands learn
ers visual hands learners visual hands learners visual hands learners visual hands learners visual
hands learners visual hands learners visual hands learners visual hands learners visual hands lear
ners visual hands learners visual hands learners visual hands learners visual hands learners visua
l hands learners visual hands learners visual hands learners visual hands learners visual hands le
arners visual hands learners visual hands learners visual hands learners visual hands learners vis
ual hands learners visual hands learners visual hands learners visual hands learners visual hands
learners visual hands learners visual hands learners visual hands learners visual hands learners v
isual hands learners visual hands learners visual hands learners visual hands learners visual hand
s learners visual hands learners visual hands learners visual hands learners visual hands learners
visual hands learners visual hands learners visual hands learners visual hands learners visual han
ds learners visual hands learners visual hands learners nannan

students eager learn also active social teach title 1 school rural arizona 75 students receive
free reduced lunch high ell population lot students never home town coming michigan hard believe n
ever seen snow student unique learning style want make sure students accommodated classroom year t
eaching math two classes second graders switch half way day two teachers mentioned earlier group d
ifferent active love move education constantly changing students needs becoming diverse sitting de
sk day not cutting lot students anymore year implemented flexible seating classroom allows student
s make choice working typically students come together carpet lesson able move working spot feel w
ould best allow work far little rugs little cushions towels students sit students also choose sit
tradition table chairs seen great results new classroom set excited keep growing classroom
everything purchased flexible seating pocket much would love able get students everything need sim
ply not realistic items project allow students physical outlet learning working times able wiggle
bounce without distracting around truly believe materials help students better learners allow meet
needs students nannan
```

Wordcloud

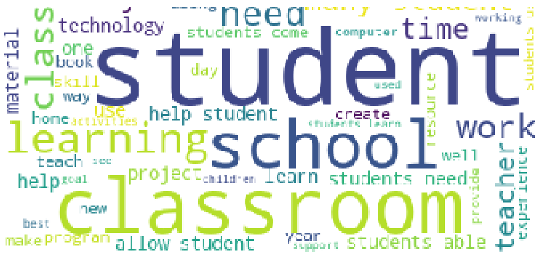
In [34]:

```python
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```



In [35]:

```python
words=''
for i in cluster2:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=50).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```



In [36]:

```
x1=bow_text(train_essay,train_title,x)
x_train = SelectKBest(f_classif, k=5000).fit_transform(x1, y_train)

x_new_train=x_train[:5000]
cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
cluster.fit_predict(x_new_train.toarray())
```

```
C:\Users\ADMIN\Anaconda3\lib\site-packages\sklearn\feature_selection\univariate_selection.py:114:
UserWarning:

Features [0 0 0 0 0 0 0 0] are constant.
```

Out[36]:

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

In [37]:

```
# For K=5

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

In [38]:

```
#Manually printing the data points
print('%s\n'%(cluster1[0]))
print('%s\n'%(cluster2[0]))
print('%s\n'%(cluster3[0]))
print('%s\n'%(cluster4[0]))
print('%s\n'%(cluster5[0]))
```

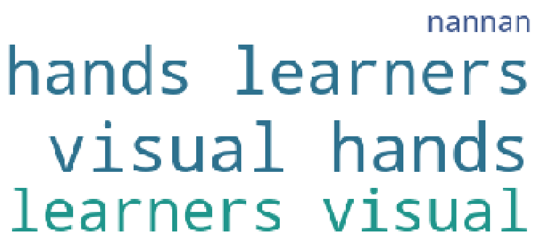
```
visual hands learners visual hands learners visual hands learners visual hands learners visual han
ds learners visual hands learners visual hands learners visual hands learners visual hands learner
s visual hands learners visual hands learners visual hands learners visual hands learners visual h
ands learners visual hands learners visual hands learners visual hands learners visual hands learn
ers visual hands learners visual hands learners visual hands learners visual hands learners visual
hands learners visual hands learners visual hands learners visual hands learners visual hands lear
ners visual hands learners visual hands learners visual hands learners visual hands learners visua
l hands learners visual hands learners visual hands learners visual hands learners visual hands le
arners visual hands learners visual hands learners visual hands learners visual hands learners vis
ual hands learners visual hands learners visual hands learners visual hands learners visual hands 
learners visual hands learners visual hands learners visual hands learners visual hands learners v
isual hands learners visual hands learners visual hands learners visual hands learners visual hand
s learners visual hands learners visual hands learners visual hands learners visual hands learners
```

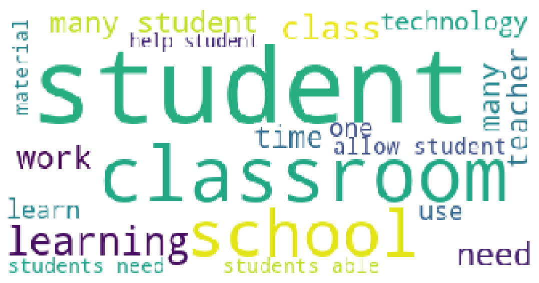
```
visual hands learners visual hands learners visual hands learners visual hands learners visual han
ds learners visual hands learners visual hands learners nannan

students eager learn also active social teach title 1 school rural arizona 75 students receive
free reduced lunch high ell population lot students never home town coming michigan hard believe n
ever seen snow student unique learning style want make sure students accommodated classroom year t
eaching math two classes second graders switch half way day two teachers mentioned earlier group d
ifferent active love move education constantly changing students needs becoming diverse sitting de
sk day not cutting lot students anymore year implemented flexible seating classroom allows student
s make choice working typically students come together carpet lesson able move working spot feel w
ould best allow work far little rugs little cushions towels students sit students also choose sit
tradition table chairs seen great results new classroom set excited keep growing classroom
everything purchased flexible seating pocket much would love able get students everything need sim
ply not realistic items project allow students physical outlet learning working times able wiggle
bounce without distracting around truly believe materials help students better learners allow meet
needs students nannan

teach mostly 9th grade students first year high school exciting confusing challenging hopefully fu
n different scenario day class try make earth science relevant possible high school diverse mixtur
e cultures races economic backgrounds ranging abject poverty comfortable 20 students receive free
reduced meals school district largely blue collar neighborhood people pull together others times n
eed remember going parents buy school supplies remember excitement kid taking everything bag organ
izing imagine come home environment resources not available able fund project benefit many ways st
udent not necessary materials class sets negative learning experience knowing come class beautiful
school supplies including notebook paper pen calculator makes student confident willing learn
addition many students suffer learning challenges adhd two balance balls intended help students ma
intain focus class numerous studies demonstrated student attention subsequent success improve
sitting balls excited see results nannan

room 18 wonderfully diverse classroom full eager learners one greatest things us friends world ame
rica europe asia learn together play together grow together look forward centers love meeting teac
her reading groups welcome homework even ask extra homework take step classroom see learning happe
ning every space quickly becoming independent learners responsible learning others great focus com
municating one another problem solving one thing learning differences make us accept not students
enjoy things understand not students learn way foundation makes us special classroom ready move 21
st century accessibility nabi allow students extend learning using academic apps math science read
ing also help students need extra stimulation electronics help focus headphones help drown sound l
earning taking place students focus work able use variety technology real world skill benefit
students need think future beyond classroom students using tablets laptops computers technology ga
dgets learn home time transition classroom space students succeed excel nannan

remember first time earned trophy ribbon not athlete imagine could reading team get rewarded showi
ng knowledge entire school district students read books join reading team compete reading battle s
tudents shining stars need little polish serve title 1 school 85 students live poverty 40 speak en
glish second language biggest obstacle access books parents cannot afford buy books often going pu
blic library involves adult able willing take school library refuge place get many materials need
free students rise challenge give sunshine state books read give reading charm collect charms enco
urage read 15 titles students get incentives along way reward parties reach goals enough copies bo
oks keep engaged difficultly offer battle book club student grades 3 5 wishes join club free read
books make detailed questions like jeopardy game students compete teams 5 answer questions correct
ly fun make friends become reading athletes school make learning look cool end school year take te
am five students club compete district wide battle books competition students chance put skills te
st incredibly bright students work team battle trophy biggest things learn sportsmanship teamwork
following rules fun important study hard proud academic achievement donation project allow offer m
ultiple copies title sunshine state list students keep interested reading learning directly impact
ing 30 students indirectly 350 core group reads books encourages others read books future
generations benefit
```

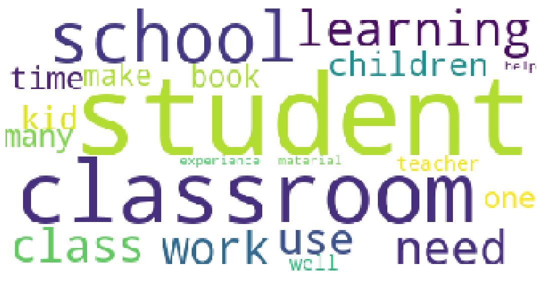
In [39]:

```python
#Wordcloud
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()

words=''
for i in cluster2:
    words+=str(i)
from wordcloud import WordCloud
```

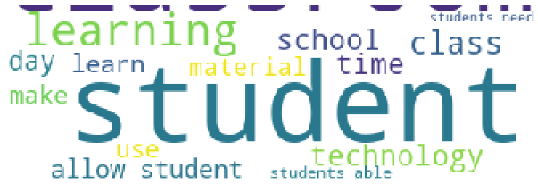
```
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()

words=''
for i in cluster3:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()

words=''
for i in cluster4:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()

words=''
for i in cluster5:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=20).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```

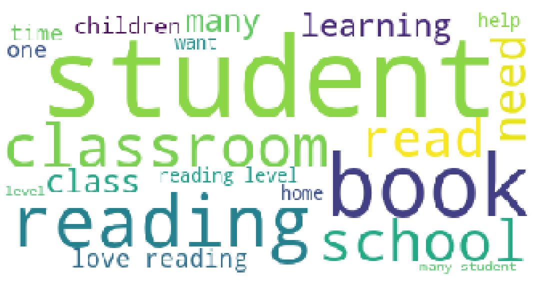






help student
create
teacher
work
need
classroom





2.7 Apply DBSCAN

2.1 Splitting data into Train and cross validation(or test): Stratified Sampling

In [40]:

```
sample_data=project_data.sample(15000)
sample_data.shape
from sklearn.model_selection import train_test_split
data=sample_data

y_train=data.project_is_approved
x_train=data.drop('project_is_approved',axis=1)



print("shape of train data ")
print(x_train.shape)
print(y_train.shape)
```

```
shape of train data
(15000, 19)
(15000,)
```

2.2 Make Data Model Ready: encoding numerical, categorical features

In [41]:

```
def veccat(x):
    from sklearn.feature_extraction.text import CountVectorizer
    from sklearn.preprocessing import Normalizer
    from scipy.sparse import hstack
    vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, binary=T
rue)
    vectorizer.fit(x['clean_categories'].values)
    print(vectorizer.get_feature_names())



    categories_one_hot = vectorizer.transform(x['clean_categories'].values)
    print("Shape of matrix after one hot encodig ",categories_one_hot.shape)

    vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
binary=True)
    vectorizer.fit(x['clean_subcategories'].values)
    print(vectorizer.get_feature_names())



    subcategories_one_hot = vectorizer.transform(x['clean_subcategories'].values)
    print("Shape of matrix after one hot encodig ",subcategories_one_hot.shape)


    vectorizer = CountVectorizer(lowercase=False, binary=True)
    vectorizer.fit(x['school_state'].values)
    print(vectorizer.get_feature_names())



    state_one_hot = vectorizer.transform(x['school_state'].values)
    print("Shape of matrix after one hot encodig ",state_one_hot.shape)

    x = x.replace(np.nan, '', regex=True)
    vectorizer = CountVectorizer( vocabulary=['Mrs.', 'Ms.','Mr.','Teacher','Dr.'],lowercase=False)
    vectorizer.fit(x['teacher_prefix'].values)
    print(vectorizer.get_feature_names())



    prefix_one_hot = vectorizer.transform(x['teacher_prefix'].values)
    print("Shape of matrix after one hot encodig ",prefix_one_hot.shape)

    vocab=[]
    for i in x['project_grade_category'].values:
        vocab.append(i)
    v_set=set(vocab)
    vocab=list(v_set)
    vectorizer = CountVectorizer(vocabulary=vocab,lowercase=False)
    vectorizer.fit(x['project_grade_category'].values)




    grade_one_hot = vectorizer.transform(x['project_grade_category'].values)
    print("Shape of matrix after one hot encodig ",grade_one_hot.shape)

    price_scalar =  Normalizer(copy=False,norm='l2')
    price_scalar.fit(x['price'].values.reshape(1,-1)) # finding the mean and standard deviation of
this data



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

    projects_scalar = Normalizer(copy=False,norm='l2')
    projects_scalar.fit(x['teacher_number_of_previously_posted_projects'].values.reshape(1,-1)) # f
inding the mean and standard deviation of this data

    # Now standardize the data with above maen and variance.
    projects_standardized =
projects_scalar.transform(x['teacher_number_of_previously_posted_projects'].values.reshape(1, -1))
    projects_standardized =np.transpose(projects_standardized)

    qty_scalar= Normalizer(copy=False,norm='l2')
    qty_scalar.fit(x['quantity'].values.reshape(1,-1)) # finding the mean and standard deviation of
this data


    # Now standardize the data with above maen and variance.
    qty_standardized = qty_scalar.transform(x['quantity'].values.reshape(1, -1))
    qty_standardized=np.transpose(qty_standardized)

    X1 = hstack((categories_one_hot, subcategories_one_hot,state_one_hot,prefix_one_hot,grade_one_h
ot, price_standardized,projects_standardized,qty_standardized))
    print(X1.shape)
    return(X1)
```

In [42]:

```
x=veccat(x_train)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'SpecialNeeds',
'Health_Sports', 'Math_Science', 'Literacy_Language']
```

```
Shape of matrix after one hot encodig  (15000, 9)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Extracurricular',
'Civics_Government', 'ForeignLanguages', 'NutritionEducation', 'Warmth', 'Care_Hunger',
'SocialSciences', 'PerformingArts', 'CharacterEducation', 'TeamSports', 'Other',
'College_CareerPrep', 'Music', 'History_Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL
', 'Gym_Fitness', 'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (15000, 30)
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', '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 after one hot encodig  (15000, 51)
['Mrs.', 'Ms.', 'Mr.', 'Teacher', 'Dr.']
Shape of matrix after one hot encodig  (15000, 5)
Shape of matrix after one hot encodig  (15000, 4)
(15000, 102)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [43]:

```
import re

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

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

In [44]:

```
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', 'e
ach', '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', "do
esn'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"]
```

In [45]:

```python
def preprocessing(x):
    import nltk
    nltk.download('stopwords')
    from tqdm import tqdm
    preprocessed_essays = []
    # tqdm is for printing the status bar
    for sentence in tqdm(x.values):
        sent = decontracted(sentence)
        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.lower() for e in sent.split() if e.lower() not in stopwords)
        preprocessed_essays.append(sent.strip())
    return preprocessed_essays
```

In [46]:

```python
train_essay=[]

train_title=[]

train_essay=preprocessing(x_train['essay'])


train_title=preprocessing(x_train['project_title'])
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|████████████████████████████████████████████████████████████████████████████| 15000/15000 [00:
18<00:00, 833.03it/s]
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
100%|██████████████████████████████████████████████████████████████████████████| 15000/15000
[00:00<00:00, 21452.54it/s]
```

Encoding using BOW

In [47]:

```python
from scipy.sparse import hstack
from sklearn.preprocessing import StandardScaler
def bow_text(train_essay,train_title,x1):
    from scipy.sparse import hstack

    vectorizer = CountVectorizer(ngram_range=(1,2),min_df=10,max_features=5000)

    vectorizer.fit(train_essay)

    text_bow=vectorizer.transform(train_essay)



    vectorizer= CountVectorizer()
    vectorizer.fit(train_title)
    title_bow=vectorizer.transform(train_title)



    x1 = hstack((x1,text_bow,title_bow )).tocsr()



    return x1
```

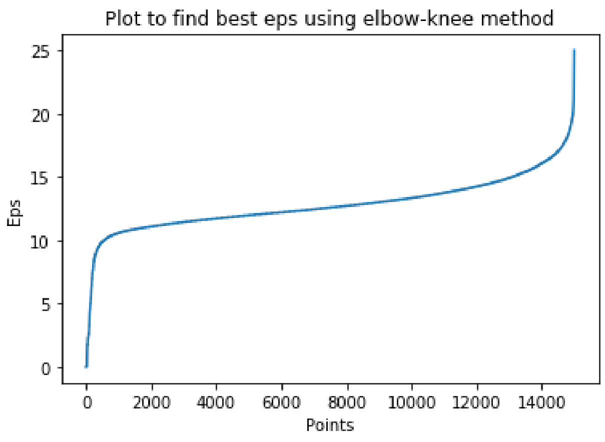
In [48]:

```python
# 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.neighbors import NearestNeighbors
from sklearn.feature_selection import SelectKBest, f_classif
x1=bow_text(train_essay,train_title,x)
x_train = SelectKBest(f_classif, k=5000).fit_transform(x1, y_train)


neigh = NearestNeighbors(n_neighbors=10000)
nbrs = neigh.fit(x_train)
distances, indices = nbrs.kneighbors(x_train)
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.plot(distances)
plt.title("Plot to find best eps using elbow-knee method")
plt.xlabel('Points')
plt.ylabel('Eps')
```

```
C:\Users\ADMIN\Anaconda3\lib\site-packages\sklearn\feature_selection\univariate_selection.py:114:
UserWarning:

Features [0 0 0 0 0 0 0 0] are constant.
```

Out[48]:

```
Text(0, 0.5, 'Eps')
```



The optimal eps is 11 as it has maximum curvature.

In [49]:

```
from sklearn.cluster import DBSCAN
m = DBSCAN(eps=11, min_samples=10000,n_jobs=-1)
m.fit(x_train)
```

Out[49]:

```
DBSCAN(algorithm='auto', eps=11, leaf_size=30, metric='euclidean',
    metric_params=None, min_samples=10000, n_jobs=-1, p=None)
```

In [50]:

```
clusters = m.labels_
print("The number of clusters is ",len(set(clusters)))
```

```
The number of clusters is  1
```

In [51]:

```
# Manually writing datapoints
cluster1 = []

for i in range(m.labels_.shape[0]):
    if m.labels_[i] == -1:
        cluster1.append(train_essay[i])
```

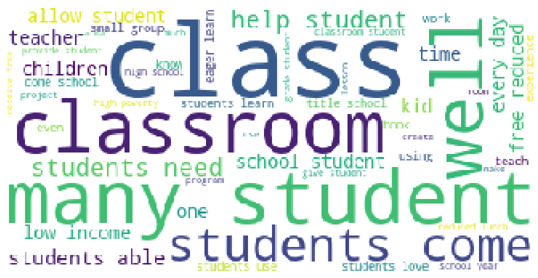
In [52]:

```
#Noise Cluster
print('%s\n'%(cluster1[0]))
```

```
ninety students enter class daily basis eager learn despite disadvantaged home environments access
technology unreliable students school non existent home access technology class regular basis impr
ove test scores college readiness students attend title 1 school provides 100 free breakfast lunch
reading writing significantly grade level however hard working students eager learn high
expectations students college bound amaze diligence positive attitude despite disadvantaged homes
e single parent low socio economic lack parental involvement testing gone digital often
intimidates college readiness skills based reading writing assessment scores given assessments
students often give not complete test easily frustrated show lower self esteem confidence using co
mputers regular basis 1 familiarize computer literacy skills needed standardized testing 2 boost s
elf confidence believe successful testing 3 help students develop college readiness skills
students use chromebooks donated practice keyboarding skills online assessments produce writing sa
mples similar found standardized testing promote college readiness skills frequent opportunities s
tudents able say yes versus no not no not comes testing thinking college students need feel confid
ent better prepared assessments order become successful school home accomplished access technology
available classroom daily basis visiting computer lab sharing ipad carts entire not enough want st
udents demonstrate perseverance testing produce researched based writing happen access technology
nannan
```

In [53]:

```
#wordcloud
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(background_color="white",max_words=50).generate(words)
# Display the generated image:
plt.imshow(wordcloud)
plt.axis("off")
plt.show()
```



3. Conclusions

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

I have used Bag of Words vectorizer in this case.

In [55]:

```
from prettytable import PrettyTable
x=PrettyTable()
x.field_names=["Vectorizer","Clustering Algorithm","Number of clusters"]
x.add_row(["BOW","Kmeans","6"])
x.add_row(["BOW","Agglomerative","2,5"])
x.add_row(["BOW","DBSCAN","1"])
print(x)
```

```
+------------+----------------------+--------------------+
| Vectorizer | Clustering Algorithm | Number of clusters |
+------------+----------------------+--------------------+
|    BOW     |        Kmeans        |         6          |
|    BOW     |    Agglomerative     |        2,5         |
|    BOW     |        DBSCAN        |         1          |
+------------+----------------------+--------------------+
```

In [ ]: