DonorsChoose

In [3]:

```
%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

import time
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
```

In [4]:

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

In [5]:

```
from sklearn.utils import resample
project_data=resample(project_data,n_samples=50000)
```

In [6]:

```
print(len(project_data))
```

```
50000
```

In [7]:

```
# how to replace elements in list python: https://stackoverflow.com/a/2582163/4084039
cols = ['Date' if x=='project_submitted_datetime' else x for x in list(project_data.columns)]
```

```
#sort dataframe based on time pandas python: https://stackoverflow.com/a/49702492/4084039
project_data['Date'] = pd.to_datetime(project_data['project_submitted_datetime'])
project_data.drop('project_submitted_datetime', axis=1, inplace=True)
project_data.sort_values(by=['Date'], inplace=True)


# how to reorder columns pandas python: https://stackoverflow.com/a/13148611/4084039
project_data = project_data[cols]

print(cols)
project_data.head(2)
```

```
['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state', 'Date',
'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']
```

Out[7]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_categ
51140	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Grades PreK-2
41558	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Grades 3-5

In [8]:

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

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

In [9]:

```
#project_data = project_data.sample(frac=0.5)
```

Preprocessing data

1.2 preprocessing of `project_subject_categories`

In [10]:

```
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"
```

```
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]))
```

In [11]:

```
preprocessed_grade=project_data['project_grade_category']
```

In [12]:

```
new=[i.replace("-","_") for i in preprocessed_grade]
new=[i.replace(" ","_") for i in new]
```

In [13]:

```
project_data['preprocessed_grade']=new
```

In [14]:

```
print(project_data['preprocessed_grade'])
```

```
0        Grades_PreK_2
1           Grades_3_5
2           Grades_3_5
3           Grades_3_5
4           Grades_3_5
5           Grades_3_5
6        Grades_PreK_2
7        Grades_PreK_2
8          Grades_9_12
9           Grades_3_5
10          Grades_3_5
11          Grades_3_5
12       Grades_PreK_2
13          Grades_3_5
14          Grades_3_5
15          Grades_3_5
16       Grades_PreK_2
17          Grades_3_5
18          Grades_3_5
19       Grades_PreK_2
20       Grades_PreK_2
21       Grades_PreK_2
22       Grades_PreK_2
23          Grades_3_5
24          Grades_6_8
25          Grades_6_8
26       Grades_PreK_2
27          Grades_6_8
28          Grades_3_5
29          Grades_3_5
             ...
49970       Grades_6_8
49971       Grades_3_5
49972       Grades_3_5
49973       Grades_3_5
49974       Grades_3_5
```

```
49975       Grades_3_5
49976       Grades_6_8
49977       Grades_3_5
49978    Grades_PreK_2
49979    Grades_PreK_2
49980    Grades_PreK_2
49981       Grades_3_5
49982       Grades_3_5
49983       Grades_3_5
49984       Grades_3_5
49985       Grades_6_8
49986    Grades_PreK_2
49987    Grades_PreK_2
49988    Grades_PreK_2
49989    Grades_PreK_2
49990    Grades_PreK_2
49991    Grades_PreK_2
49992    Grades_PreK_2
49993       Grades_6_8
49994       Grades_6_8
49995       Grades_3_5
49996    Grades_PreK_2
49997       Grades_3_5
49998    Grades_PreK_2
49999    Grades_PreK_2
Name: preprocessed_grade, Length: 50000, dtype: object
```

In [15]:

```
print(project_data['clean_categories'].unique())
```

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

1.3 preprocessing of `project_subject_subcategories`

In [16]:

```
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)

# 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 of project_grade_category

1.3 Text preprocessing

In [17]:

```
# 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 [18]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

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

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

In [19]:

```
# 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', '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 [20]:

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

```
100%|██████████████████████████████████████████████████████████████████████| 50000/50000
[00:35<00:00, 1390.63it/s]
```

In [21]:

```
project_data['project_resource_summary']
preprocessed_resource_summary=[]
for sentence in tqdm(project_data['project_resource_summary'].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 not in stopwords)
    preprocessed_resource_summary.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████| 50000/50000
[00:03<00:00, 12788.28it/s]
```

In [22]:

```
project_data['clean_resource_summary'] = preprocessed_resource_summary
```

1.4 Preprocessing of `project_title`

In [23]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_titles = []
# tqdm is for printing the status bar
for sentence in tqdm(project_data['project_title'].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 for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:01<00:00, 29193.16it/s]
```

In [24]:

```
#Adding processed columns at place of original columns
project_data['clean_essays'] = preprocessed_essays
#project_data.drop(['project_essay_1'], axis=1, inplace=True)
#project_data.drop(['project_essay_2'], axis=1, inplace=True)
#project_data.drop(['project_essay_3'], axis=1, inplace=True)
#project_data.drop(['project_essay_4'], axis=1, inplace=True)
```

In [25]:

```
project_data['clean_titles'] = preprocessed_titles
```

In [26]:

```
# we cannot remove rows where teacher prefix is not available therefore we are replacing 'nan' val
ue with
# 'null'(string)
#https://stackoverflow.com/questions/42224700/attributeerror-float-object-has-no-attribute-split
project_data['teacher_prefix'] = project_data['teacher_prefix'].fillna('null')
```

In [27]:

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

Out[27]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_category
0	74477	p189804	4a97f3a390bfe21b99cf5e2b81981c73	Mrs.	CA	2016-04-27 00:46:53	Grades PreK-2
1	33679	p137682	06f6e62e17de34fcf81020c77549e1d5	Mrs.	WA	2016-04-27 01:05:25	Grades 3-5

2 rows × 24 columns

In [28]:

```
project_data.count()
```

Out[28]:

```
Unnamed: 0                                        50000
id                                                50000
teacher_id                                        50000
teacher_prefix                                    50000
school_state                                      50000
Date                                              50000
project_grade_category                            50000
```

```
project_grade_category                          50000
project_title                                   50000
project_essay_1                                 50000
project_essay_2                                 50000
project_essay_3                                  1709
project_essay_4                                  1709
project_resource_summary                        50000
teacher_number_of_previously_posted_projects    50000
project_is_approved                             50000
price                                           50000
quantity                                        50000
clean_categories                                50000
preprocessed_grade                              50000
clean_subcategories                             50000
essay                                           50000
clean_resource_summary                          50000
clean_essays                                    50000
clean_titles                                    50000
dtype: int64
```

So far we have preprocessed the data. Next is to split and vectorize data for BoW,TFIDF,Avg W2Vec and TFIDF weighted W2Vec

1.Splitting data

In [31]:

```
y = project_data['project_is_approved'].values
project_data.drop(['project_is_approved'], axis=1, inplace=True)
X = project_data
```

In [32]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

In [33]:

```
x = np.count_nonzero(y_test)
print(len(y_test) - x)
```

```
2528
```

In [34]:

```
print(X_train.shape, y_train.shape)
print(X_cv.shape, y_cv.shape)
print(X_test.shape, y_test.shape)

print("="*100)
```

```
(22445, 23) (22445,)
(11055, 23) (11055,)
(16500, 23) (16500,)
====================================================================================================
```

2.Vectorizing data

BoW

2.1 Text data

In [35]:

In [35]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10, max_features=5000)
vectorizer.fit(X_train['clean_essays'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_essay_bow = vectorizer.transform(X_train['clean_essays'].values)
X_cv_essay_bow = vectorizer.transform(X_cv['clean_essays'].values)
X_test_essay_bow = vectorizer.transform(X_test['clean_essays'].values)

print("After vectorizations")
print(X_train_essay_bow.shape, y_train.shape)
print(X_cv_essay_bow.shape, y_cv.shape)
print(X_test_essay_bow.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 5000) (22445,)
(11055, 5000) (11055,)
(16500, 5000) (16500,)
====================================================================================================
```

In [36]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10, max_features=5000)
vectorizer.fit(X_train['clean_titles'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_titles_bow = vectorizer.transform(X_train['clean_titles'].values)
X_cv_titles_bow = vectorizer.transform(X_cv['clean_titles'].values)
X_test_titles_bow = vectorizer.transform(X_test['clean_titles'].values)

print("After vectorizations")
print(X_train_titles_bow.shape, y_train.shape)
print(X_cv_titles_bow.shape, y_cv.shape)
print(X_test_titles_bow.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1232) (22445,)
(11055, 1232) (11055,)
(16500, 1232) (16500,)
====================================================================================================
```

In [37]:

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(min_df=10, max_features=5000)
vectorizer.fit(X_train['clean_resource_summary'].values) # fit has to happen only on train data
#Bow_FeatureList=Bow_FeatureList + (vectorizer.get_feature_names())
# we use the fitted CountVectorizer to convert the text to vector
X_train_summary_bow = vectorizer.transform(X_train['clean_resource_summary'].values)
X_cv_summary_bow = vectorizer.transform(X_cv['clean_resource_summary'].values)
X_test_summary_bow = vectorizer.transform(X_test['clean_resource_summary'].values)

print("After vectorizations")
print(X_train_summary_bow.shape, y_train.shape)
print(X_cv_summary_bow.shape, y_cv.shape)
print(X_test_summary_bow.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 2498) (22445,)
(11055, 2498) (11055,)
(16500, 2498) (16500,)
====================================================================================================
```

In [38]:

```
X_train_summary_bow.shape
```

Out[38]:

```
(22445, 2498)
```

2.2 one hot encoding the catogorical features: clean_categories

In [39]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_categories'].values) # fit has to happen only on train data


# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_cat_ohe = vectorizer.transform(X_train['clean_categories'].values)
X_cv_clean_cat_ohe = vectorizer.transform(X_cv['clean_categories'].values)
X_test_clean_cat_ohe = vectorizer.transform(X_test['clean_categories'].values)

print("After vectorizations")
print(X_train_clean_cat_ohe.shape, y_train.shape)
print(X_cv_clean_cat_ohe.shape, y_cv.shape)
print(X_test_clean_cat_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 9) (22445,)
(11055, 9) (11055,)
(16500, 9) (16500,)
['appliedlearning', 'care_hunger', 'health_sports', 'history_civics', 'literacy_language',
'math_science', 'music_arts', 'specialneeds', 'warmth']
====================================================================================================
```

2.3 one hot encoding the catogorical features: clean_subcategories

In [40]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['clean_subcategories'].values) # fit has to happen only on train data


# we use the fitted CountVectorizer to convert the text to vector
X_train_clean_subcat_ohe = vectorizer.transform(X_train['clean_subcategories'].values)
X_cv_clean_subcat_ohe = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_clean_subcat_ohe = vectorizer.transform(X_test['clean_subcategories'].values)

print("After vectorizations")
print(X_train_clean_subcat_ohe.shape, y_train.shape)
print(X_cv_clean_subcat_ohe.shape, y_cv.shape)
print(X_test_clean_subcat_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 30) (22445,)
(11055, 30) (11055,)
(16500, 30) (16500,)
['appliedsciences', 'care_hunger', 'charactereducation', 'civics_government',
'college_careerprep', 'communityservice', 'earlydevelopment', 'economics', 'environmentalscience',
'esl', 'extracurricular', 'financialliteracy', 'foreignlanguages', 'gym_fitness',
'health_lifescience', 'health_wellness', 'history_geography', 'literacy', 'literature_writing', 'm
athematics', 'music', 'nutritioneducation', 'other', 'parentinvolvement', 'performingarts', 'socia
lsciences', 'specialneeds', 'teamsports', 'visualarts', 'warmth']
====================================================================================================
```

2.3 one hot encoding the catogorical features: teacher_prefix

In [41]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_teacher_ohe = vectorizer.transform(X_train['teacher_prefix'].values)
X_cv_teacher_ohe = vectorizer.transform(X_cv['teacher_prefix'].values)
X_test_teacher_ohe = vectorizer.transform(X_test['teacher_prefix'].values)

print("After vectorizations")
print(X_train_teacher_ohe.shape, y_train.shape)
print(X_cv_teacher_ohe.shape, y_cv.shape)
print(X_test_teacher_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 6) (22445,)
(11055, 6) (11055,)
(16500, 6) (16500,)
['dr', 'mr', 'mrs', 'ms', 'null', 'teacher']
====================================================================================================
```

2.4 one hot encoding the catogorical features: school_state

In [42]:

```
vectorizer = CountVectorizer()
vectorizer.fit(X_train['school_state'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_state_ohe = vectorizer.transform(X_train['school_state'].values)
X_cv_state_ohe = vectorizer.transform(X_cv['school_state'].values)
X_test_state_ohe = vectorizer.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_train_state_ohe.shape, y_train.shape)
print(X_cv_state_ohe.shape, y_cv.shape)
print(X_test_state_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 51) (22445,)
(11055, 51) (11055,)
(16500, 51) (16500,)
['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']
====================================================================================================
```

2.4 one hot encoding the catogorical features: project_grade_category

In [43]:

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

Out[43]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_cate

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_cate
33555	107910	p047706	c8abebb59c5fe865f8e9a41be4997aca	Mrs.	TX	2016-12-06 09:34:55	Grades 3-5
45280	57493	p138655	5b787d56cf5325351a34eff2aeb97c2d	Mrs.	WA	2017-03-16 15:54:59	Grades 3-5

2 rows × 23 columns

In [44]:

```
#This step is to intialize a vectorizer with vocab from train data
from collections import Counter
my_counter4 = Counter()
for word in X_train['preprocessed_grade'].values:
    my_counter4.update(word.split())

# dict sort by value python: https://stackoverflow.com/a/613218/4084039
project_grade_category_dict = dict(my_counter4)
sorted_project_grade_category_dict = dict(sorted(project_grade_category_dict.items(), key=lambda
kv: kv[1]))
```

In [45]:

```
vectorizer = CountVectorizer(vocabulary=list(sorted_project_grade_category_dict.keys()), lowercase
=False, binary=True)
vectorizer.fit(X_train['preprocessed_grade'].values) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_train_grade_ohe = vectorizer.transform(X_train['preprocessed_grade'].values)
X_cv_grade_ohe = vectorizer.transform(X_cv['preprocessed_grade'].values)
X_test_grade_ohe = vectorizer.transform(X_test['preprocessed_grade'].values)

print("After vectorizations")
print(X_train_grade_ohe.shape, y_train.shape)
print(X_cv_grade_ohe.shape, y_cv.shape)
print(X_test_grade_ohe.shape, y_test.shape)
print(vectorizer.get_feature_names())
print("="*100)
```

```
After vectorizations
(22445, 4) (22445,)
(11055, 4) (11055,)
(16500, 4) (16500,)
['Grades_9_12', 'Grades_6_8', 'Grades_3_5', 'Grades_PreK_2']
====================================================================================================
```

2.5 Normalizing the numerical features: Price

In [46]:

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

Out[46]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_cate
33555	107910	p047706	c8abebb59c5fe865f8e9a41be4997aca	Mrs.	TX	2016-12-06 09:34:55	Grades 3-5

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	Date	project_grade_cate
45280	57493	p138655	5b787d56cf5325351a34eff2aeb97c2d	Mrs.	WA	2017-03-16 15:54:59	Grades 3-5

2 rows × 23 columns

In [47]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# 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.
standard_vec.fit(X_train['price'].values.reshape(-1,1))

X_train_price_std = standard_vec.transform(X_train['price'].values.reshape(-1,1))
X_cv_price_std = standard_vec.transform(X_cv['price'].values.reshape(-1,1))
X_test_price_std = standard_vec.transform(X_test['price'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_price_std.shape, y_train.shape)
print(X_cv_price_std.shape, y_cv.shape)
print(X_test_price_std.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

2.6 Vectorizing numerical features: teacher_number_of_previously _posted_projects"

In [48]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# 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.
standard_vec.fit(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1))

X_train_projects_std =
standard_vec.transform(X_train['teacher_number_of_previously_posted_projects'].values.reshape(-1,1
))
X_cv_projects_std = standard_vec.transform(X_cv['teacher_number_of_previously_posted_projects'].va
lues.reshape(-1,1))
X_test_projects_std = standard_vec.transform(X_test['teacher_number_of_previously_posted_projects'
].values.reshape(-1,1))

print("After vectorizations")
print(X_train_projects_std.shape, y_train.shape)
print(X_cv_projects_std.shape, y_cv.shape)
print(X_test_projects_std.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

In [49]:

```
from sklearn.preprocessing import StandardScaler
standard_vec = StandardScaler(with_mean = False)
# 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.
standard_vec.fit(X_train['quantity'].values.reshape(-1,1))

X_train_qty_std = standard_vec.transform(X_train['quantity'].values.reshape(-1,1))
X_cv_qty_std = standard_vec.transform(X_cv['quantity'].values.reshape(-1,1))
X_test_qty_std = standard_vec.transform(X_test['quantity'].values.reshape(-1,1))

print("After vectorizations")
print(X_train_qty_std.shape, y_train.shape)
print(X_cv_qty_std.shape, y_cv.shape)
print(X_test_qty_std.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(22445, 1) (22445,)
(11055, 1) (11055,)
(16500, 1) (16500,)
====================================================================================================
```

In [50]:

```
#function to get heatmap confusion matrix for obtaining colur encoded matrix
#Reference link https://seaborn.pydata.org/generated/seaborn.heatmap.html
def get_confusion_matrix(clf,X_te,y_test):
    y_pred = clf.predict(X_te)
    df_cm = pd.DataFrame(confusion_matrix(y_test, y_pred), range(2),range(2))
    df_cm.columns = ['Predicted NO','Predicted YES']
    df_cm = df_cm.rename({0: 'Actual NO', 1: 'Actual YES'})
    sns.set(font_scale=1.4)#for label size
    sns.heatmap(df_cm, annot=True,annot_kws={"size": 16}, fmt='g')
```

2.7 Concatinating all the features

In [51]:

```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack
X_tr =
hstack((X_train_essay_bow,X_train_titles_bow,X_train_summary_bow,X_train_clean_cat_ohe,X_train_clea
n_subcat_ohe, X_train_state_ohe, X_train_teacher_ohe, X_train_grade_ohe,
X_train_price_std,X_train_projects_std,X_train_qty_std)).tocsr()
X_cr =
hstack((X_cv_essay_bow,X_cv_titles_bow,X_cv_summary_bow,X_cv_clean_cat_ohe,X_cv_clean_subcat_ohe,
X_cv_state_ohe, X_cv_teacher_ohe, X_cv_grade_ohe, X_cv_price_std,X_cv_projects_std,X_cv_qty_std)).t
ocsr()
X_te =
hstack((X_test_essay_bow,X_test_titles_bow,X_test_summary_bow,X_test_clean_cat_ohe,X_test_clean_sub
cat_ohe, X_test_state_ohe, X_test_teacher_ohe, X_test_grade_ohe,
X_test_price_std,X_test_projects_std,X_test_qty_std)).tocsr()

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

```
Final Data matrix
(22445, 8833) (22445,)
(11055, 8833) (11055,)
(16500, 8833) (16500,)
```

==========

Clustering

2.4 Dimensionality Reduction on the selected features

In [52]:

```
from sklearn.feature_selection import SelectKBest, chi2
t = SelectKBest(chi2, k=5000).fit(X_tr, y_train)
X_tr = t. transform(X_tr)
X_te = t. transform(X_te)

print(X_tr.shape)
```

```
(22445, 5000)
```

In [53]:

```
inertia=[]
```

In [89]:

```
from sklearn.cluster import KMeans
#n_clusters =[3,4,5,6,7]
n_clusters =[3,4,5,6,7,8,9,10,11,12]
for i in n_clusters:
    kmeans =KMeans(n_clusters=i, n_jobs=-1).fit(X_new)
    inertia . append(kmeans.inertia_)
```

In [90]:

```
inertia
```

Out[90]:

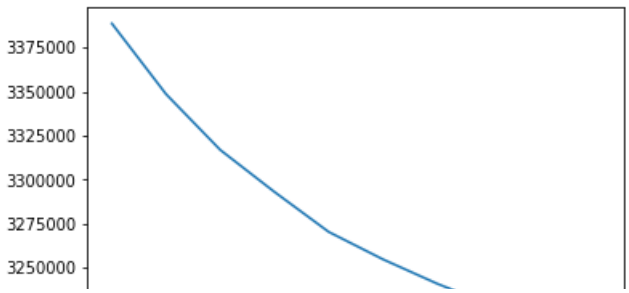
```
[3388593.3200988714,
 3348484.058290435,
 3316817.833367131,
 3292973.11482428,
 3270285.797129399,
 3254719.7453254997,
 3240903.5731392954,
 3228674.066961946,
 3217513.572754767,
 3208218.0958067332]
```

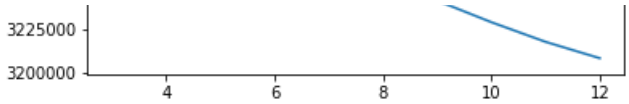
In [91]:

```
plt.plot([3,4,5,6,7,8,9,10,11,12],inertia, label='Elbow plot b/w K and inertia')
```

Out[91]:

```
[<matplotlib.lines.Line2D at 0x25193e19f60>]
```





Optimal K value

In [55]:

```
from sklearn.cluster import KMeans
optimal_k =7
kmeans =KMeans(n_clusters=optimal_k, n_jobs=-1).fit(X_tr)
```

In [59]:

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

cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []
cluster6 = []
cluster7 = []
for i in range(kmeans.labels_.shape[0]):
    if kmeans.labels_[i] == 0:
        cluster1.append(essays[i])
    elif kmeans.labels_[i] == 1:
        cluster2.append(essays[i])
    elif kmeans.labels_[i] == 2:
        cluster3.append(essays[i])
    elif  kmeans.labels_[i] == 3:
        cluster4.append(essays[i])
    elif  kmeans.labels_[i] == 4:
        cluster5.append(essays[i])
    elif  kmeans.labels_[i] == 5:
        cluster6.append(essays[i])
    elif  kmeans.labels_[i] == 6:
        cluster7.append(essays[i])
```

In [60]:

```
for i in range(2):
    print(cluster2[i])
    print("\n")
```

```
students today face many challenges teach school high percentage students live poverty many
students face significant struggles home yet everyday walk school happy ready learn mixture
students struggle gifted students students adhd students special ed needs day work hard work toget
her keep great attitude feel lucky group kiddos year classroom using flexible seating flexible sea
ting allows student choose spot room work feel comfortable able complete best work storage supplie
s important order flexible seating successful supplies one central location easily accessible stud
ents important wood designs storage unit classroom organized supplies handy kids not enough storag
e supplies right would storage unit would awesome room nannan


students third graders colorado diverse students many different cultures family values special way
different skills showcase every day hope help students feel confident work excel academically goal
also teach good citizens positive outlook towards learning deserve best learning environment possi
ble seating pads allow students sit floor reading circle time comfortably also serve cushion
chairs sit instruction time students need move little fantastic microphone help students able hear
voice even volume feel like students front classroom hearing voice loudly students back not hear w
ell enough microphone would even whole class students hear clearly nannan
```

In [61]:

```
for i in range(2):
    print(cluster4[i])
    print("\n")
```

```
teaching poorest big city united states stated u census bureau students faced many challenges many
adults not experience life time goal create classroom environment excites students moment walk doo
r much motivated learn want come school day african american woman grew low income community servi
ng african american students low income community see reflection know better tools resources class
room positive academic outcomes scholars well create classroom culture fosters differentiated lear
ning among students kindergarten always exciting time teachers parents scholars first real experie
nce public school system bunch kindergarten students bold creative energetic projects give strong
academic foundation resulting positive outlook education every scholar different individual
learning styles contribute unique educational skill sets amazon fire kids tablet allow students ac
cess technology serve research tool intervention resource extension activity expeditionary
learning school students conduct research books field work online sources donate project allow off
er scholars differentiated approaches learning enabling connect relate understand academic concept
s kindergarten students comprehension literacy skills serve foundation learning key success
learning expeditions amazon fire kids tablet allow students access audio books literacy games educ
ational videos help kindergarten scholars research world around way relates technological world na
nnan


700 students school joy work upbeat cheerful demonstrate good character visit media center learn f
un engaging ways believe hands exploration tool students benefits importantly students demonstrate
desire take risks learning demonstrate growth mindset active learners willing try new things due f
eel going ready anything future holds store project media center would like acquire five sets
bloxels bloxels interactive building sets allow children build design capture image tablet utilize
app build video game based design video game app allows build designs game share knowledge others
bloxel website shares many lessons plans help students learn review content gaming engaging
interests would like help fourth fifth grade students increase content knowledge better way learn
topics like digital citizenship share research findings game fully engaged students remember conte
nt better learning crucial technology coding skills prepare future learning careers thank
considering project nannan
```

In [62]:

```
for i in range(2):
    print(cluster6[i])
    print("\n")
```

```
great group students three four years multi age classrooms students part family students come diff
erent socioeconomic backgrounds many qualifying free milk school place feel safe loved learn try c
reate warm comfortable atmosphere fosters love learning every year students need non conventional
seating better lighting students enthusiastic learning especially math reading science not like ri
gid conventional seats would like provide alternative seating like bean bag chairs stability
cushions book gift enjoyed cd player allow students listen books tape cd want students healthy ess
ential oils help students many ways peppermint helps students stay alert really great test days la
vender balsam help create stress free relaxed atmosphere thieves helps killing germs essential
oils healthy alternative chemicals reading workshop meet small groups 4 5 students swivel stool he
lp work students organized classroom makes easier students self sufficient instills sense pride ca
rt tz tape meant help organize materials students access independently needed digital timers help
students monitor time centers foam dice help students play math games not disturbing rest class wo
rking centers need lamp reading nook help create cozy atmosphere students prefer ambient light ins
tead fluorescent lighting nannan


determined provide amazing students high quality 21st century college prep education not want chal
lenges teaching urban public school resourced community eclipse many joys occur students set succe
ss school located poorest congressional district country not stop students coming school high hope
s future justice sonia sotomayor grandmaster flash also came neighborhood stopping students
dreaming big despite sometimes enormous obstacles home communities students full good eager cultur
ally relevant instruction texts speak lived realities learning activities kindle sparks
intellectual rigor order achieve need access technology facilitates meaningful engaged learning vo
ltaire said ear avenue heart could added brain students need 12 headphones would help cultivate
personalized instruction students individual needs met specific ways sony headphones many uses str
uggling readers would able read accompaniment audiobook literacy software class school extra help
sessions would able tune distractions around order concentrate reading students would able view li
sten supplemental videos tie particular lesson ted talks documentary films resources online headph
ones classroom would especially give visual auditory learners access curriculum one way invite tee
nagers meaningful learning make authentic engaged multimedia headphones would provide struggling
students access audiobooks videos music original project creations year english class students cre
ate exhibitions interdisciplinary projects based certain topic investigation students could use he
adphones work projects create podcasts presentation voiceovers see hear help providing class headp
hones open students ears minds exciting new avenues literacy learning nannan
```

In [63]:

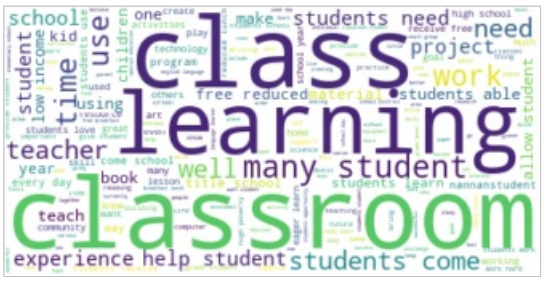
```
from wordcloud import WordCloud
def print_wordcloud(cluster):
    words=''
    for i in cluster:
        words+=str(i)

    wordcloud = WordCloud(background_color="white").generate(words)

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

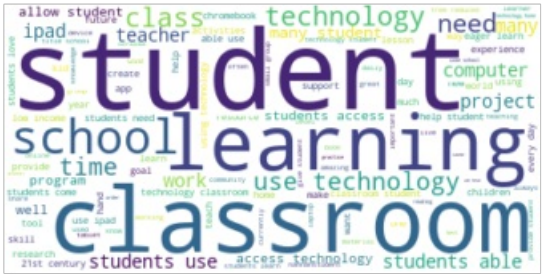
In [64]:

```
print_wordcloud(cluster1)
```



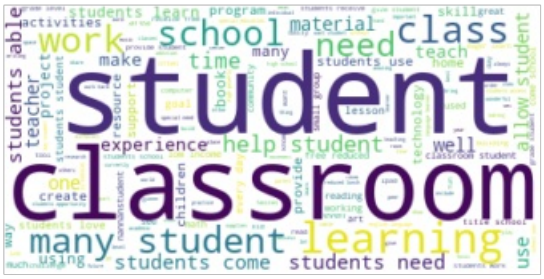
In [65]:

```
print_wordcloud(cluster3)
```



In [66]:

```
print_wordcloud(cluster6)
```



AgglomerativeClustering

In [67]:

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

In [68]:

```
from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', linkage='ward')
aggl = cluster.fit(X_tr.toarray() )
```

In [71]:

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

cluster1 = []
cluster2 = []

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

In [72]:

```
for i in range(2):
    print(cluster1[i])
    print("\n")
```

```
students magnificent group energetic learners array disabilities diverse group whose ages range 6
12 attend title school represent neighborhood live 15 males 9 females enthusiasm despite
disabilities inspiration eagerness approach learning day daunting reason seeking assistance helpin
g maintain level enthusiasm increased academic performance providing resource demonstrated success
materials make difference students learning improve school lives helping stay focused task many st
udents diagnosed adhd academically limiting disability easily distracted miss great deal quality c
ritical instruction improved academic performance help build confidence confidence help accelerate
learning process look forward successful use materials experience rewarding students school lives
greatly enhanced successful use wiggle chairs nannan


students walk class every day eagerness learn strive every day give students best learning
opportunities always willing give 100 learning many students come homes low poverty drive
education amazes group 5th graders always goes beyond expected try fifth graders write big famous
person report every year books would really help research portion report students write reports dr
ess like person entire school year students picked people teddy roosevelt barrack obama marie curr
ie name books would really help research chosen popular people also people students not chose woul
d great able classroom use researching writing papers nannan
```

In [73]:

```
for i in range(2):
    print(cluster2[i])
    print("\n")
```

```
students hard workers represent diversity community classroom reflects families together speak fiv
e different languages students enjoy learning reading writing math social studies science love han
ds projects learn communication skills language environment conducive learning every day big deal
learn best goal make students love learning make school fun place leveled books needed help studen
ts learn read leveled book help early readers read books appropriate reading stages reading books
level help students advance next reading level become skilled confident readers leveled books help
kindergarten students improve readers students read books appropriate reading ability interest dev
elop confident readers learning read essential exciting also scary challenging students read books
interest poses appropriate challenge reading ability students want read leveled books one factors
contribute making learning read fun kindergarteners need leveled books read school well take home
confidently practice reading skills books provide appropriate challenge reader well interest stude
nts gain confidence current level advance next level learning read fun exciting build confidence e
arly reader nannan
```

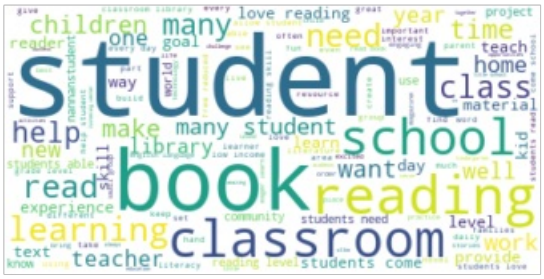
```
teach title school 95 students qualify free reduced cost lunch 80 students go home non english spe
aking families students english speaking people family make goal every year supply students suppli
es need hands experiences would not get elsewhere love students finish activity say great time muc
h learned students bright deserve every opportunity get every year state florida chooses fifteen e
xcellent written books awarded sunshine state young reader award fifteen ssyra books announced
many elementary schools across state hold contests third fourth fifth graders see read fifteen boo
ks many books student read county actually brings actual voting booths student reads three books g
ets vote favorite book love reading love idea ssyra books love encouraging students read read read
nannan
```

In [74]:

```
print_wordcloud(cluster1)
```

In [75]:

```
print_wordcloud(cluster2)
```



In [76]:

```
from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n_clusters=5, affinity='euclidean', linkage='ward')
aggl = cluster.fit(X_tr . toarray())
```

In [77]:

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

cluster1 = []
cluster2 = []

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

In [78]:

```
for i in range(2):
    print(cluster1[i])
```

```
    print(cluster1[i])
    print("\n")
```

```
students hard workers represent diversity community classroom reflects families together speak fiv
e different languages students enjoy learning reading writing math social studies science love han
ds projects learn communication skills language environment conducive learning every day big deal
learn best goal make students love learning make school fun place leveled books needed help studen
ts learn read leveled book help early readers read books appropriate reading stages reading books
level help students advance next reading level become skilled confident readers leveled books help
kindergarten students improve readers students read books appropriate reading ability interest dev
elop confident readers learning read essential exciting also scary challenging students read books
interest poses appropriate challenge reading ability students want read leveled books one factors
contribute making learning read fun kindergarteners need leveled books read school well take home
confidently practice reading skills books provide appropriate challenge reader well interest stude
nts gain confidence current level advance next level learning read fun exciting build confidence e
arly reader nannan


teach title school 95 students qualify free reduced cost lunch 80 students go home non english spe
aking families students english speaking people family make goal every year supply students suppli
es need hands experiences would not get elsewhere love students finish activity say great time muc
h learned students bright deserve every opportunity get every year state florida chooses fifteen e
xcellent written books awarded sunshine state young reader award fifteen ssyra books announced
many elementary schools across state hold contests third fourth fifth graders see read fifteen boo
ks many books student read county actually brings actual voting booths student reads three books g
ets vote favorite book love reading love idea ssyra books love encouraging students read read read
nannan
```

In [79]:

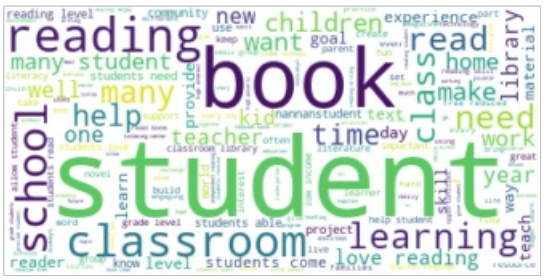
```
for i in range(2):
    print(cluster2[i])
    print("\n")
```

```
students today face many challenges teach school high percentage students live poverty many
students face significant struggles home yet everyday walk school happy ready learn mixture
students struggle gifted students students adhd students special ed needs day work hard work toget
her keep great attitude feel lucky group kiddos year classroom using flexible seating flexible sea
ting allows student choose spot room work feel comfortable able complete best work storage supplie
s important order flexible seating successful supplies one central location easily accessible stud
ents important wood designs storage unit classroom organized supplies handy kids not enough storag
e supplies right would storage unit would awesome room nannan


students third graders colorado diverse students many different cultures family values special way
different skills showcase every day hope help students feel confident work excel academically goal
also teach good citizens positive outlook towards learning deserve best learning environment possi
ble seating pads allow students sit floor reading circle time comfortably also serve cushion
chairs sit instruction time students need move little fantastic microphone help students able hear
voice even volume feel like students front classroom hearing voice loudly students back not hear w
ell enough microphone would even whole class students hear clearly nannan
```

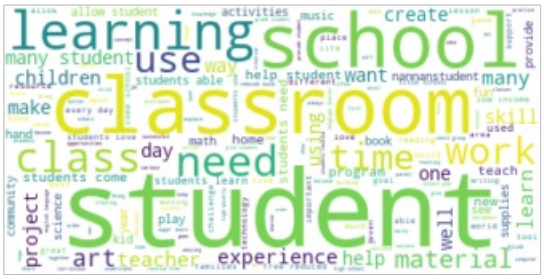
In [80]:

```
print_wordcloud(cluster1)
```



In [81]:

```
print_wordcloud(cluster2)
```



DBSCAN

In [87]:

```
from sklearn.metrics.pairwise import euclidean_distances
X_dense = X_tr .toarray()
dist= euclidean_distances(X_dense , X_dense [1].reshape(1, -1))
```

In [88]:

```
dist.shape
```

Out[88]:

```
(5000, 1)
```

In [89]:

```
dist
```

Out[89]:

```
array([[12.71543492],
       [ 0.        ],
       [13.30288981],
       ...,
       [17.09515036],
       [17.08552004],
       [16.09482523]])
```

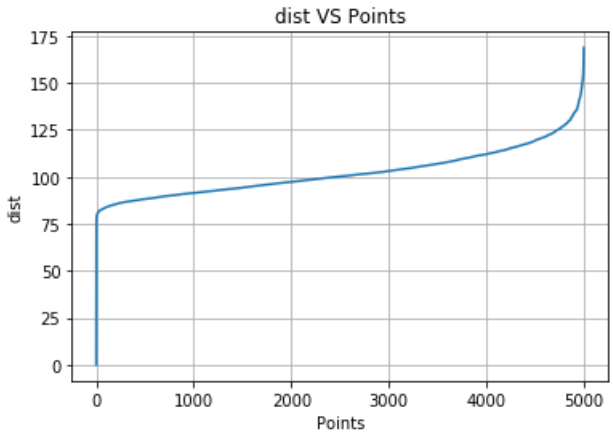
In [91]:

```
#Reference https://github.com/dileepteja3/Clustering-on-Donors-
choose/blob/master/dileep.teja3%40gmail.com_10.ipynb
X_std=StandardScaler().fit_transform(X_dense)
min_points =1000
distance=[]
for point in tqdm(X_std):
    temp = euclidean_distances(X_std, point.reshape(1, -1))
    distance.append(temp[min_points])
sorted_distance = np.sort(np.array(distance))

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


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

```
100%|██████████████████████████████████████████████████████████████████████| 5000/5000 [05
:19<00:00, 15.55it/s]
```



In [92]:

```
from sklearn.cluster import DBSCAN

clustering=DBSCAN(eps=81 , n_jobs=-1) .fit(X_std)

print('No of clusters: ',len(set(clustering.labels_)))
print('Cluster are including noise i.e -1: ',set(clustering.labels_))
```

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

In [95]:

```
cluster1=[]
noisecluster1=[]
for i in range(clustering.labels_.shape[0]):
    if clustering.labels_[i] == 0:
        cluster1.append(essays[i])
    elif clustering.labels_[i] == -1:
        noisecluster1.append(essays[i])
```

In [96]:

```
for i in range(2):
    print(cluster1[i])
    print("\n")
```

```
students magnificent group energetic learners array disabilities diverse group whose ages range 6
12 attend title school represent neighborhood live 15 males 9 females enthusiasm despite
disabilities inspiration eagerness approach learning day daunting reason seeking assistance helpin
g maintain level enthusiasm increased academic performance providing resource demonstrated success
materials make difference students learning improve school lives helping stay focused task many st
udents diagnosed adhd academically limiting disability easily distracted miss great deal quality c
ritical instruction improved academic performance help build confidence confidence help accelerate
learning process look forward successful use materials experience rewarding students school lives
greatly enhanced successful use wiggle chairs nannan


students walk class every day eagerness learn strive every day give students best learning
opportunities always willing give 100 learning many students come homes low poverty drive
education amazes group 5th graders always goes beyond expected try fifth graders write big famous
person report every year books would really help research portion report students write reports dr
ess like person entire school year students picked people teddy roosevelt barrack obama marie curr
ie name books would really help research chosen popular people also people students not chose woul
d great able classroom use researching writing papers nannan
```

In [97]:

In [97]:

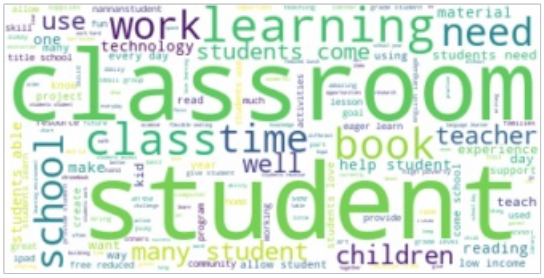
```
for i in range(2):
    print(noisecluster1[i])
    print("\n")
```

```
students today face many challenges teach school high percentage students live poverty many
students face significant struggles home yet everyday walk school happy ready learn mixture
students struggle gifted students students adhd students special ed needs day work hard work toget
her keep great attitude feel lucky group kiddos year classroom using flexible seating flexible sea
ting allows student choose spot room work feel comfortable able complete best work storage supplie
s important order flexible seating successful supplies one central location easily accessible stud
ents important wood designs storage unit classroom organized supplies handy kids not enough storag
e supplies right would storage unit would awesome room nannan


determined provide amazing students high quality 21st century college prep education not want chal
lenges teaching urban public school resourced community eclipse many joys occur students set succe
ss school located poorest congressional district country not stop students coming school high hope
s future justice sonia sotomayor grandmaster flash also came neighborhood stopping students
dreaming big despite sometimes enormous obstacles home communities students full good eager cultur
ally relevant instruction texts speak lived realities learning activities kindle sparks
intellectual rigor order achieve need access technology facilitates meaningful engaged learning vo
ltaire said ear avenue heart could added brain students need 12 headphones would help cultivate
personalized instruction students individual needs met specific ways sony headphones many uses str
uggling readers would able read accompaniment audiobook literacy software class school extra help
sessions would able tune distractions around order concentrate reading students would able view li
sten supplemental videos tie particular lesson ted talks documentary films resources online headph
ones classroom would especially give visual auditory learners access curriculum one way invite tee
nagers meaningful learning make authentic engaged multimedia headphones would provide struggling
students access audiobooks videos music original project creations year english class students cre
ate exhibitions interdisciplinary projects based certain topic investigation students could use he
adphones work projects create podcasts presentation voiceovers see hear help providing class headp
hones open students ears minds exciting new avenues literacy learning nannan
```

In [98]:

```
print_wordcloud(cluster1)
```



In [99]:

```
print_wordcloud(noisecluster1)
```