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	
`project_id`	A unique identifier for the proposed project
`project_title`	Title of tl • Art Wil •
`project_grade_category`	Grade level of students for which the project is targete • • • •

Feature	
project_subject_categories	One or more (comma-separated) subject categories f following enur • • • • • Li • • • • • • Literacy & Languag
school_state	State where school is located (Two- (https://en.wikipedia.org/wiki/List_of_U.S._state_abbrevia
project_subject_subcategories	One or more (comma-separated) subject subcate • • Literature & Writing
project_resource_summary	An explanation of the resources needed for t • My students need hands on literacy ma sen
project_essay_1	F
project_essay_2	Sec
project_essay_3	Tl
project_essay_4	Fou
project_submitted_datetime	Datetime when project application was submitted. **Ex**
teacher_id	A unique identifier for the teacher of the propo bdf8baa8fedef6b
teacher_prefix	Teacher's title. One of the following • • • • • •
teacher_number_of_previously_posted_projects	Number of project applications previously submitte

* 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_4:__ "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
```

1.1 Reading Data

In [2]:
```
project_data = pd.read_csv('train_data.csv',nrows=50000)
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 (50000, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'scho
ol_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
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [5]:
```
project_data["project_is_approved"].value_counts()
```

Out[5]:
```
1    42286
0     7714
Name: project_is_approved, dtype: int64
```

1.2 preprocessing of project_subject_categories

In [6]:
```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.co

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-
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",
        if 'The' in j.split(): # this will split each of the catogory based on sp
            j=j.replace('The','') # if we have the words "The" we are going to re
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trail
        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 [7]:
```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.co

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

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",
        if 'The' in j.split(): # this will split each of the catogory based on sp
            j=j.replace('The','') # if we have the words "The" we are going to re
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty)
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trail
        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/22898595/4
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]))
```

1.2.7 Univariate Analysis: Text features (Project Essay's)

In [8]:
```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

1.2.8 Univariate Analysis: Cost per project

In [9]:
```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[9]:

	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

In [10]:
```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).
price_data.head(2)
```

Out[10]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [11]:
```
# join two dataframes in python:
project_data = pd.merge(project_data, price_data, on='id', how='left')

print(project_data)
```

```
       Unnamed: 0       id                         teacher_id teacher_prefix
\
0          160221  p253737  c90749f5d961ff158d4b4d1e7dc665fc           Mrs.
1          140945  p258326  897464ce9ddc600bced1151f324dd63a            Mr.
2           21895  p182444  3465aaf82da834c0582ebd0ef8040ca0            Ms.
3              45  p246581  f3cb9bffbba169bef1a77b243e620b60           Mrs.
4          172407  p104768  be1f7507a41f8479dc06f047086a39ec           Mrs.
5          141660  p154343  a50a390e8327a95b77b9e495b58b9a6e           Mrs.
6           21147  p099819  9b40170bfa65e399981717ee8731efc3           Mrs.
7           94142  p092424  5bfd3d12fae3d2fe88684bbac570c9d2            Ms.
8          112489  p045029  487448f5226005d08d36bdd75f095b31           Mrs.
9          158561  p001713  140eeac1885c820ad5592a409a3a8994            Ms.
10          43184  p040307  363788b51d40d978fe276bcb1f8a2b35           Mrs.
11         127083  p251806  4ba7c721133ef651ca54a03551746708            Ms.
12          19090  p051126  5e52c92b7e3c472aad247a239d345543           Mrs.
13          15126  p003874  178f6ae765cd4e0fb143a77c47fd65e2           Mrs.
14          62232  p233127  424819801de22a60bba7d0f4354d0258            Ms.
15          67303  p132832  bb6d6d054824fa01576ab38dfa2be160            Ms.
16         127215  p174627  4ad7e280fddff889e1355cc9f29c3b89           Mrs.
```

In [12]:
```
approved_price = project_data[project_data['project_is_approved']==1]['price'].va
print
rejected_price = project_data[project_data['project_is_approved']==0]['price'].va
```

1.3 Text preprocessing

1.3.1 Essay Text

In [13]:
```
project_data.head(2)
```

Out[13]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20

In [14]:
```
# printing some random essays.
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
```

```
My students are English learners that are working on English as their second or
third languages. We are a melting pot of refugees, immigrants, and native-born
Americans bringing the gift of language to our school. \r\n\r\n We have over 24
languages represented in our English Learner program with students at every lev
el of mastery.  We also have over 40 countries represented with the families wi
thin our school.  Each student brings a wealth of knowledge and experiences to
us that open our eyes to new cultures, beliefs, and respect.\"The limits of you
r language are the limits of your world.\"-Ludwig Wittgenstein  Our English lea
rner's have a strong support system at home that begs for more resources.  Many
times our parents are learning to read and speak English along side of their ch
ildren.  Sometimes this creates barriers for parents to be able to help their c
hild learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy p
roviding these dvd's and players, students are able to continue their mastery o
f the English language even if no one at home is able to assist.  All families
with students within the Level 1 proficiency status, will be a offered to be a
part of this program.  These educational videos will be specially chosen by the
English Learner Teacher and will be sent home regularly to watch.  The videos a
re to help the child develop early reading skills.\r\n\r\nParents that do not h
ave access to a dvd player will have the opportunity to check out a dvd player
to use for the year.  The plan is to use these videos and educational dvd's for
the years to come for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year all
love learning, at least most of the time. At our school, 97.3% of the students
receive free or reduced price lunch. Of the 560 students, 97.3% are minority st
udents. \r\nThe school has a vibrant community that loves to get together and c
elebrate. Around Halloween there is a whole school parade to show off the beaut
iful costumes that students wear. On Cinco de Mayo we put on a big festival wit
h crafts made by the students, dances, and games. At the end of the year the sc
hool hosts a carnival to celebrate the hard work put in during the school year,
with a dunk tank being the most popular activity.My students will use these fiv
e brightly colored Hokki stools in place of regular, stationary, 4-legged chair
s. As I will only have a total of ten in the classroom and not enough for each
student to have an individual one, they will be used in a variety of ways. Duri
ng independent reading time they will be used as special chairs students will e
ach use on occasion. I will utilize them in place of chairs at my small group t
ables during math and reading times. The rest of the day they will be used by t
he students who need the highest amount of movement in their life in order to s
tay focused on school.\r\n\r\nWhenever asked what the classroom is missing, my
students always say more Hokki Stools. They can't get their fill of the 5 stool
s we already have. When the students are sitting in group with me on the Hokki
Stools, they are always moving, but at the same time doing their work. Anytime
the students get to pick where they can sit, the Hokki Stools are the first to
be taken. There are always students who head over to the kidney table to get on
e of the stools who are disappointed as there are not enough of them. \r\n\r\nW
e ask a lot of students to sit for 7 hours a day. The Hokki stools will be a co
mpromise that allow my students to do desk work and move at the same time. Thes
e stools will help students to meet their 60 minutes a day of movement by allow
ing them to activate their core muscles for balance while they sit. For many of
```

```
my students, these chairs will take away the barrier that exists in schools for
a child who can't sit still.nannan
==================================================
```

In [15]:
```
# 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 [16]:
```
sent = decontracted(project_data['essay'].values[4000])
print(sent)
print("="*50)
```

```
I teach language arts and social studies to about 50 students each day.  I teac
h two groups of amazing kids each day!\r\n\r\nThe students in my classroom rang
e from advanced or gifted learners to students with various learning disabiliti
es. My school is located in an urban environment in Maryland. The school is a T
itle I (low-income) school, and 99% of the students in the school receive free
and reduced price lunch. All students at my school receive free breakfast which
is the most important meal of the day!High interest reading supports comprehens
ion and learning. I want to encourage a love of reading by choosing books that
interest my third grade students. Many of my students are classified as \"strug
gling readers\". There is extensive research to support the premise that the be
st way to become a better reader is to read more. In order for my students to b
ecome better or more fluent readers I need to increase both the quantity and qu
ality of their reading.  They need reading materials that they can read and wil
l want to read. \r\n\r\nI want to send my students into summer vacation with a
high interest book. If they find success and interest with one book, research s
hows that learning will generate more learning! The book I have chosen is reada
ble, has a convincing plot, and has realistic characters.nannan
==================================================
```

In [17]:
```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-bre
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
I teach language arts and social studies to about 50 students each day.  I teac
h two groups of amazing kids each day!    The students in my classroom range fr
om advanced or gifted learners to students with various learning disabilities.
My school is located in an urban environment in Maryland. The school is a Title
I (low-income) school, and 99% of the students in the school receive free and r
educed price lunch. All students at my school receive free breakfast which is t
he most important meal of the day!High interest reading supports comprehension
and learning. I want to encourage a love of reading by choosing books that inte
rest my third grade students. Many of my students are classified as  struggling
readers . There is extensive research to support the premise that the best way
to become a better reader is to read more. In order for my students to become b
etter or more fluent readers I need to increase both the quantity and quality o
f their reading.  They need reading materials that they can read and will want
to read.     I want to send my students into summer vacation with a high intere
st book. If they find success and interest with one book, research shows that l
earning will generate more learning! The book I have chosen is readable, has a
convincing plot, and has realistic characters.nannan
```

In [18]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
I teach language arts and social studies to about 50 students each day I teach
two groups of amazing kids each day The students in my classroom range from adv
anced or gifted learners to students with various learning disabilities My scho
ol is located in an urban environment in Maryland The school is a Title I low i
ncome school and 99 of the students in the school receive free and reduced pric
e lunch All students at my school receive free breakfast which is the most impo
rtant meal of the day High interest reading supports comprehension and learning
I want to encourage a love of reading by choosing books that interest my third
grade students Many of my students are classified as struggling readers There i
s extensive research to support the premise that the best way to become a bette
r reader is to read more In order for my students to become better or more flue
nt readers I need to increase both the quantity and quality of their reading Th
ey need reading materials that they can read and will want to read I want to se
nd my students into summer vacation with a high interest book If they find succ
ess and interest with one book research shows that learning will generate more
learning The book I have chosen is readable has a convincing plot and has reali
stic characters nannan
```

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'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he',
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itse
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'tha
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'ha
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 't
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'of
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than'
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should'v
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "d
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma'
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn
            'won', "won't", 'wouldn', "wouldn't"]
```

In [20]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|
| 50000/50000 [01:11<00:00, 697.16it/s]
```

In [21]:
```
# after preprocesing
preprocessed_essays[2000]

project_data['essay']=pd.DataFrame(preprocessed_essays)
```

1.3.2 Project title Text

In [22]:
```
# similarly you can preprocess the titles also

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

```
100%|██████████████████████████████████████████████████████████████████████████████|
50000/50000 [00:04<00:00, 12227.39it/s]
```

In [23]:
```
preprocessed_titles[2000]

project_data['project_title']=pd.DataFrame(preprocessed_titles)
```

1. 4 Preparing data for models

In [24]:
```
project_data.columns
```

Out[24]:
```
Index(['Unnamed: 0', 'id', 'teacher_id', 'teacher_prefix', 'school_state',
       'project_submitted_datetime', 'project_grade_category', 'project_title',
       'project_essay_1', 'project_essay_2', 'project_essay_3',
       'project_essay_4', 'project_resource_summary',
       'teacher_number_of_previously_posted_projects', 'project_is_approved',
       'clean_categories', 'clean_subcategories', 'essay', 'price',
       'quantity'],
      dtype='object')
```

we are going to consider

- school_state : categorical data
- clean_categories : categorical data
- clean_subcategories : categorical data
- project_grade_category : categorical data
- teacher_prefix : categorical data

- project_title : text data
- text : text data
- project_resource_summary: text data

- quantity : numerical
- teacher_number_of_previously_posted_projects : numerical
- price : numerical

In [25]:
```
grades = list(project_data['project_grade_category'].values)
# remove special characters from list of strings python: https://stackoverflow.co

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-
grades_list = []
for i in grades:
    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",
        j = j.replace(' ','_') # we are placeing all the ' '(space) with ''(empty
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trail
        temp = temp.replace('-','_') # we are replacing the & value into
        temp = temp.replace('Grades','grades') # we are replacing the & value int
        temp = temp.replace('PreK','prek') # we are replacing the & value into
    grades_list.append(temp.strip())

project_data['project_grade_category'] = grades_list
```

Assignment 10: Clustering

- step 1: Choose any vectorizer (data matrix) that you have worked in any of the assignments, and got the best AUC value.
- step 2: Choose any of the feature selection (https://scikit-learn.org/stable/modules/feature_selection.html)/reduction algorithms (https://scikit-learn.org/stable/modules/decomposition.html) ex: selectkbest features, pretrained word vectors, model based feature selection etc and reduce the number of features to 5k features
- step 3: Apply all three kmeans, Agglomerative clustering, DBSCAN
  - **K-Means Clustering:**
    - Find the best 'k' using the elbow-knee method (plot k vs inertia_)
  - **Agglomerative Clustering:**
    - Apply agglomerative algorithm (https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/) and try a different number of clusters like 2,5 etc.
    - You can take less data points (as this is very computationally expensive one) to perform hierarchical clustering because they do take a considerable amount of time to run.
  - **DBSCAN Clustering:**
    - Find the best 'eps' using the elbow-knee method (https://stackoverflow.com/a/48558030/4084039).
    - You can take a smaller sample size for this as well.
- step 4: Summarize each cluster by manually observing few points from each cluster.
- step 5: You need to plot the word cloud with essay text for each cluster for each of algorithms mentioned in step 3.

2. Clustering

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

In [26]:
```
X=project_data
```

In [27]:
```
X.head()
```

Out[27]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_sul
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	20
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	20
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	20
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	20
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	20

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

2.2 Make Data Model Ready: encoding numerical, categorical features

Normalizing the numerical features: Price

In [28]:
```
from sklearn.preprocessing import Normalizer
normalizer_price = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
normalizer_price.fit(X['price'].values.reshape(-1,1))
X_price_norm = normalizer_price.transform(X['price'].values.reshape(-1,1))
print("After vectorizations")
print(X_price_norm.shape)
print("="*100)
```

```
After vectorizations
(50000, 1)
====================================================================================================
```

Normalizing the numerical features: Previously posted projects

In [29]:
```
from sklearn.preprocessing import Normalizer
normalizer_ppp = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
normalizer_ppp.fit(X['teacher_number_of_previously_posted_projects'].values.resha
X_ppp_norm = normalizer_ppp.transform(X['teacher_number_of_previously_posted_pro
print("After vectorizations")
print(X_ppp_norm.shape)
print("="*100)
```

```
After vectorizations
(50000, 1)
====================================================================================================
```

Normalizing the numerical features : Quantity

In [30]:
```
from sklearn.preprocessing import Normalizer
normalizer_qty = Normalizer()
# normalizer.fit(X_train['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96 96.01 ... 368.98 80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1) if it contains a single sample.
normalizer_qty.fit(X['quantity'].values.reshape(-1,1))
X_qty_norm = normalizer_qty.transform(X['quantity'].values.reshape(-1,1))

print("After vectorizations")

print(X_qty_norm.shape)

print("="*100)
```

```
After vectorizations
(50000, 1)
====================================================================================================
```

One hot encoding the catogorical features: State

In [31]:
```
vectorizer_schoolstate = CountVectorizer(lowercase=False, binary=True)
vectorizer_schoolstate.fit(X['school_state'].values) # fit has to happen only on

# we use the fitted CountVectorizer to convert the text to vector
X_state_ohe = vectorizer_schoolstate.transform(X['school_state'].values)

print("After vectorizations")
print(X_state_ohe.shape)

print(vectorizer_schoolstate.get_feature_names())
print("="*100)

stateVec=vectorizer_schoolstate.get_feature_names()
type(stateVec)
```

```
After vectorizations
(50000, 51)
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA',
'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS',
'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'OR', 'PA',
'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV', 'WY']
====================================================================================================
```

Out[31]:
```
list
```

One hot encoding the catogorical features: Project Grade

In [32]:
```
uniqueData=X['project_grade_category'].unique()
print(uniqueData)


vectorizer_grade = CountVectorizer()
vectorizer_grade.fit(X['project_grade_category'].values) # fit has to happen only

# we use the fitted CountVectorizer to convert the text to vector
X_grade_ohe = vectorizer_grade.transform(X['project_grade_category'].values)

print("After vectorizations")
print(X_grade_ohe.shape)

print(vectorizer_grade.get_feature_names())
print("="*100)

projGradeVec=vectorizer_grade.get_feature_names()
```

```
['grades_prek_2' 'grades_6_8' 'grades_3_5' 'grades_9_12']
After vectorizations
(50000, 4)
['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
====================================================================================
======================
```

One hot encoding the catogorical features: Teacher Prefix

In [33]:
```
#replacing nan with empty string
X.teacher_prefix=X.teacher_prefix.fillna('')

uniqueData=X['teacher_prefix'].unique()
print(uniqueData)

vectorizer_teachprefix = CountVectorizer(lowercase=False, binary=True)
vectorizer_teachprefix.fit(X['teacher_prefix'].values) # fit has to happen only

# we use the fitted CountVectorizer to convert the text to vector
X_teacher_ohe = vectorizer_teachprefix.transform(X['teacher_prefix'].values)

print("After vectorizations")
print(X_teacher_ohe.shape)

print(vectorizer_teachprefix.get_feature_names())
print("="*100)

prefixteacherVec=vectorizer_teachprefix.get_feature_names()
```

```
['Mrs.' 'Mr.' 'Ms.' 'Teacher' '' 'Dr.']
After vectorizations
(50000, 5)
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
====================================================================================
======================
```

One hot encoding the catogorical features: Clean categories

In [34]:
```
vectorizer_cleancat = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), l
vectorizer_cleancat.fit(X['clean_categories'].values) # fit has to happen only o

# we use the fitted CountVectorizer to convert the text to vector
X_ccat_ohe = vectorizer_cleancat.transform(X['clean_categories'].values)

print("After vectorizations")
print(X_ccat_ohe.shape)

print(vectorizer_cleancat.get_feature_names())
print("="*100)

cleanCatVec=vectorizer_cleancat.get_feature_names()
```

```
After vectorizations
(50000, 9)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'S
pecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
====================================================================================
======================
```

One hot encoding the catogorical features: Cleab subcategories

In [35]:
```
vectorizer_cleansubcat = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys
vectorizer_cleansubcat.fit(X['clean_subcategories'].values) # fit has to happen o

# we use the fitted CountVectorizer to convert the text to vector
X_csub_ohe = vectorizer_cleansubcat.transform(X['clean_subcategories'].values)

print("After vectorizations")
print(X_csub_ohe.shape)

print(vectorizer_cleansubcat.get_feature_names())
print("="*100)

cleansubCatVec=vectorizer_cleansubcat.get_feature_names()
```

```
After vectorizations
(50000, 30)
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement', 'Ex
tracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducation',
'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducatio
n', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_Geography',
'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness', 'EnvironmentalS
cience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'L
iterature_Writing', 'Mathematics', 'Literacy']
====================================================================================
====================
```

2.3 Make Data Model Ready: encoding essay, and project_title

Bag of Words

In [36]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_bow = CountVectorizer(min_df=10,ngram_range=(1,4))
vectorizer_bow.fit(X['project_title']) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_title_bow = vectorizer_bow.transform(X['project_title'])

print("After vectorizations")
print(X_title_bow.shape)

print("="*100)

projTitleBowVec=vectorizer_bow.get_feature_names()
```

```
After vectorizations
(50000, 4160)
====================================================================================
====================
```

In [37]:
```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer_bow = CountVectorizer(min_df=10,ngram_range=(1,4))
vectorizer_bow.fit(X['essay']) # fit has to happen only on train data

# we use the fitted CountVectorizer to convert the text to vector
X_essay_bow = vectorizer_bow.transform(X['essay'])

print("After vectorizations")
print(X_essay_bow.shape)

print("="*100)

projTitleBowVec=vectorizer_bow.get_feature_names()
```

```
After vectorizations
(50000, 169756)
====================================================================================================
```

In [39]:
```
# merge two sparse matrices: https://stackoverflow.com/a/19710648/4084039
from scipy.sparse import hstack

X_bow=hstack((X_price_norm,X_ppp_norm,X_qty_norm,X_state_ohe,X_grade_ohe,X_teache

print("After vectorizer")

print(X_bow.shape)
```

```
After vectorizer
(50000, 174018)
```

In [40]:
```
y=project_data['project_is_approved'].values
```

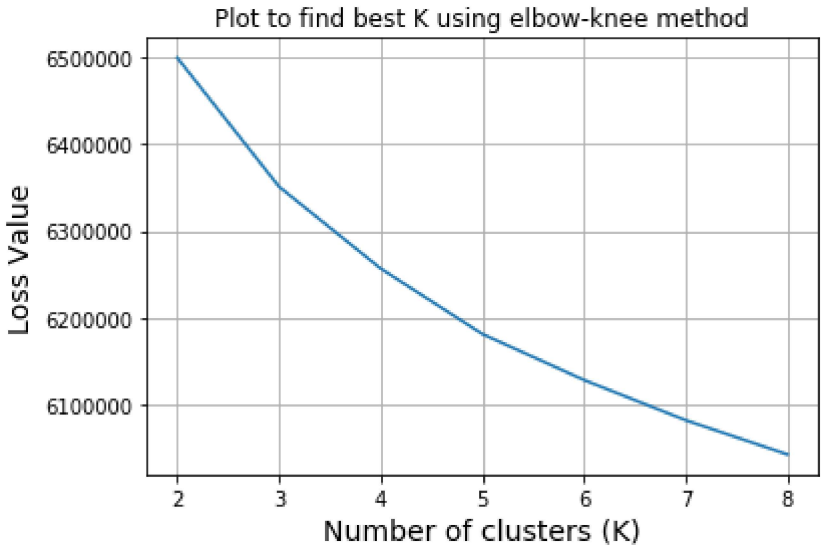
2.4 Dimensionality Reduction on the selected features

In [40]:
```
# using selectKBest we are find top 5k features


from sklearn.feature_selection import SelectKBest, chi2
t = SelectKBest(chi2,k=5000).fit(X_bow,y)
X_new = t.transform(X_bow)

print("="*100)

print("Final Data matrix on Bag of words")
print(X_new.shape)

print("="*100)
```

```
====================================================================================================
Final Data matrix on Bag of words
(50000, 5000)
====================================================================================================
```

2.5 Apply Kmeans

In [41]:
```
from sklearn.cluster import KMeans
k_values = [2,3,4,5,6,7,8]
loss = []
for i in k_values:
    kmeans = KMeans(n_clusters=i, random_state=0, n_jobs=-1).fit(X_new)
    loss.append(kmeans.inertia_)
plt.plot(k_values, loss)
plt.xlabel('Number of clusters (K)',size=14)
plt.ylabel('Loss Value',size=14)
plt.title("Plot to find best K using elbow-knee method")
plt.grid()
plt.show()
```



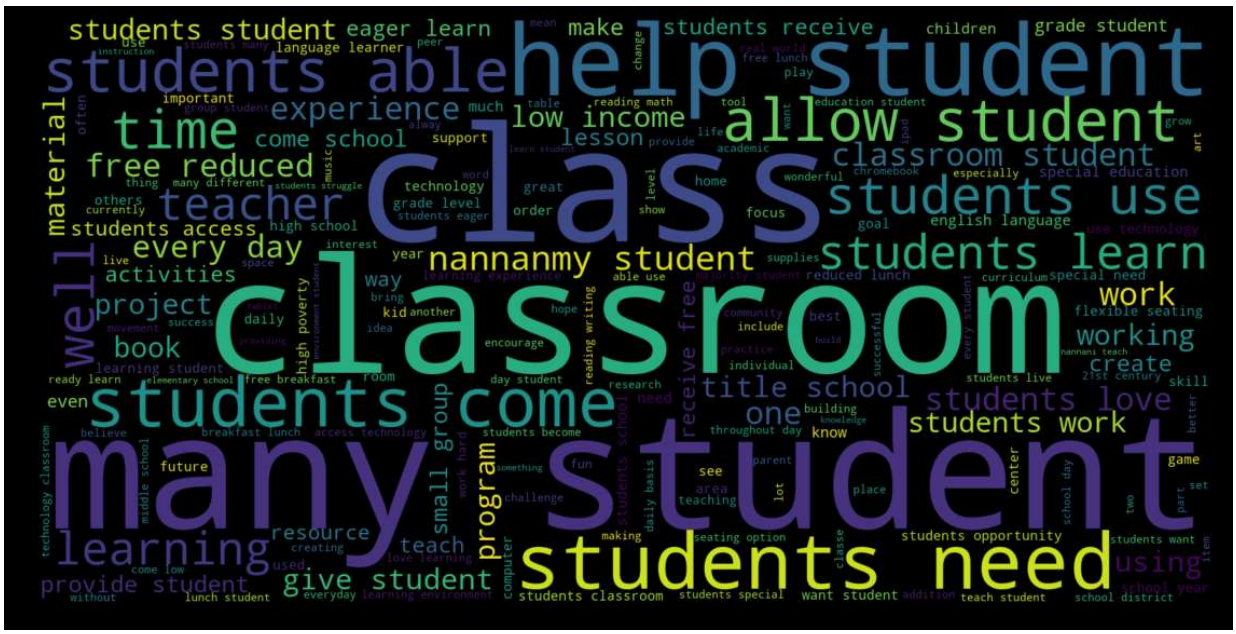
In [42]:
```
optimal_k = 6

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

In [43]:
```
print(kmeans.labels_.shape)
```

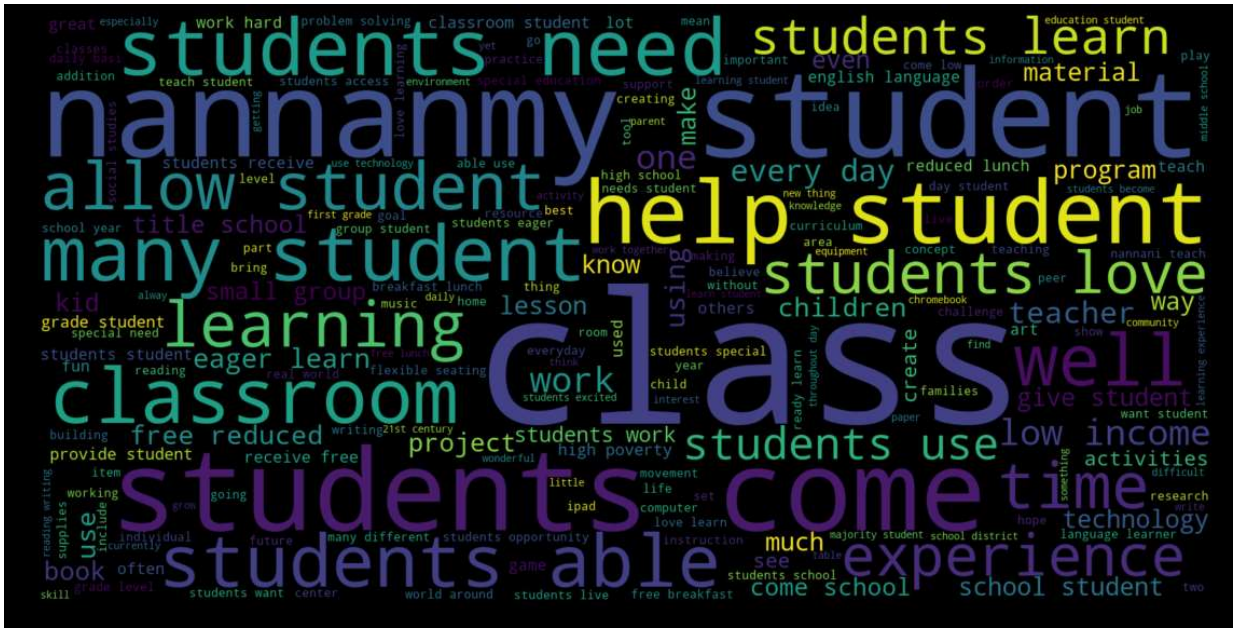
```
(50000,)
```

In [44]:
```
essays = X['essay'].values

cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []
cluster6 = []


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

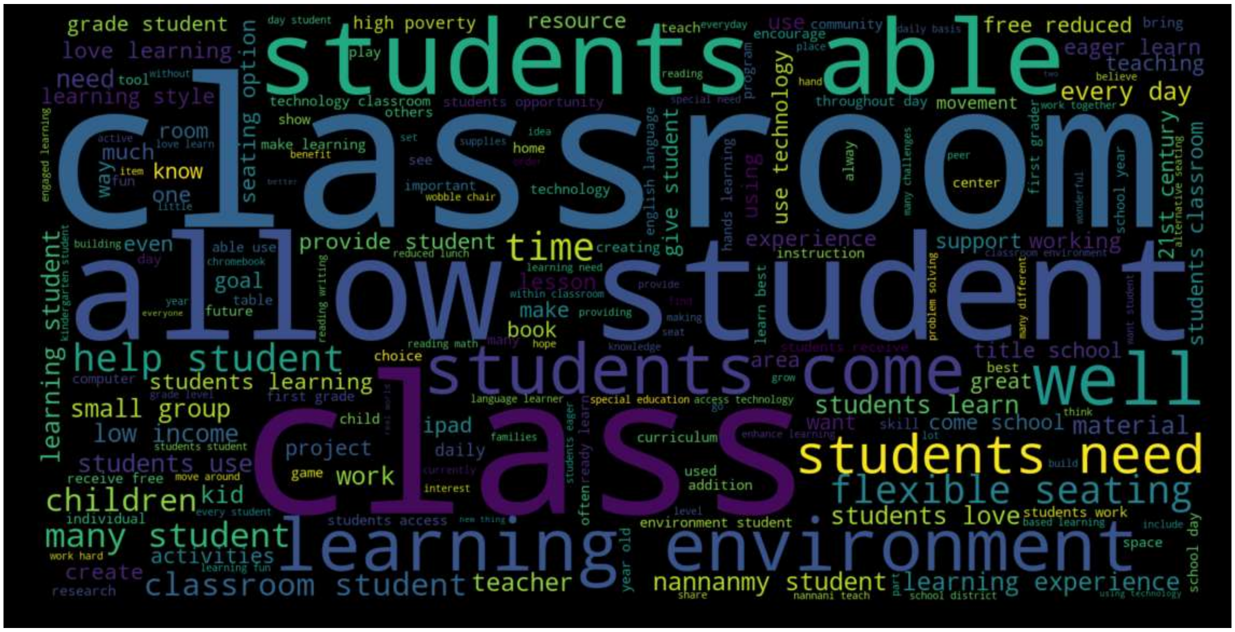
In [45]:
```
#cluster 1
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



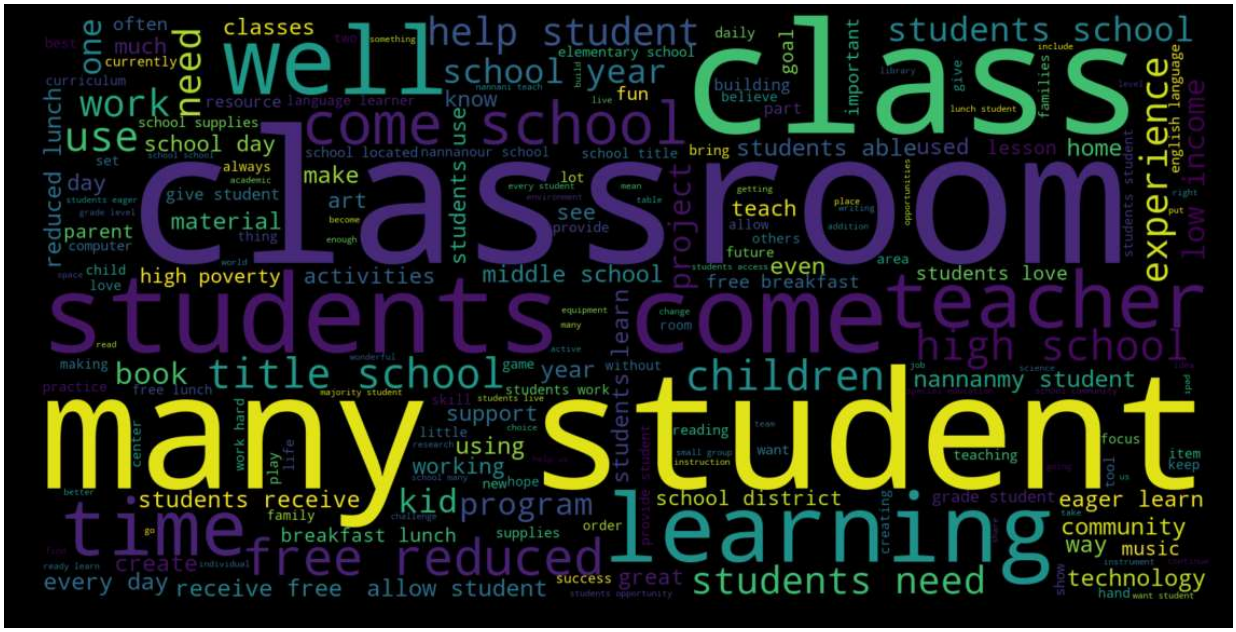
In [46]:
```
#cluster 2
words=''
for i in cluster2:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



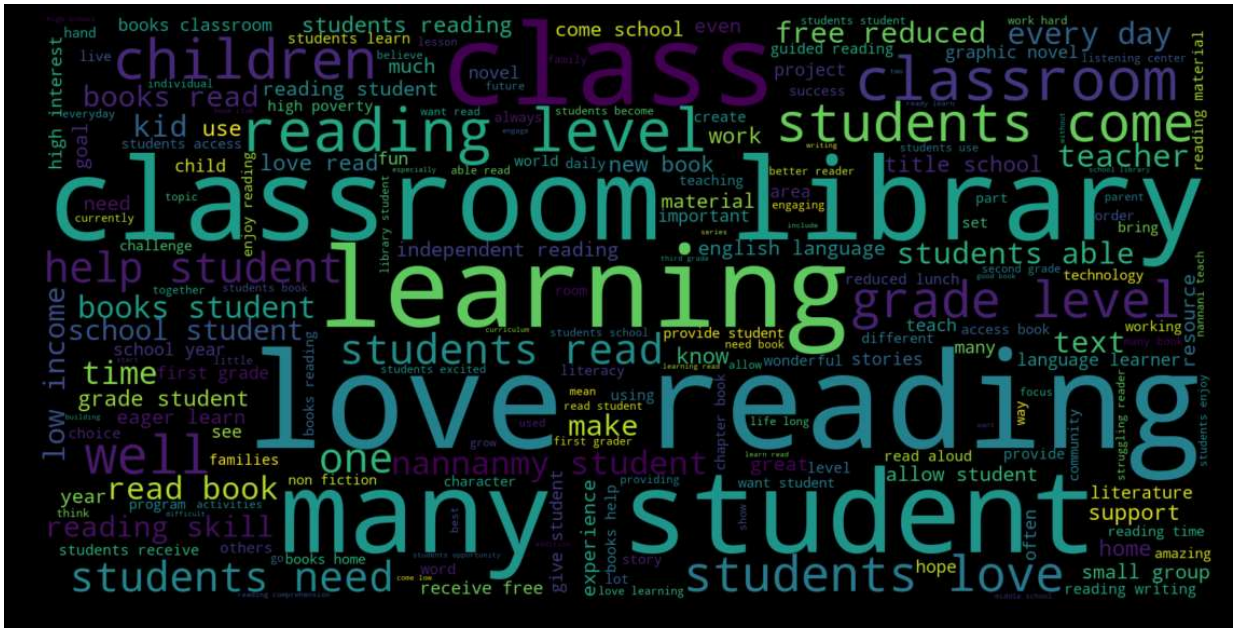
In [47]:
```
#cluster 3
words=''
for i in cluster3:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



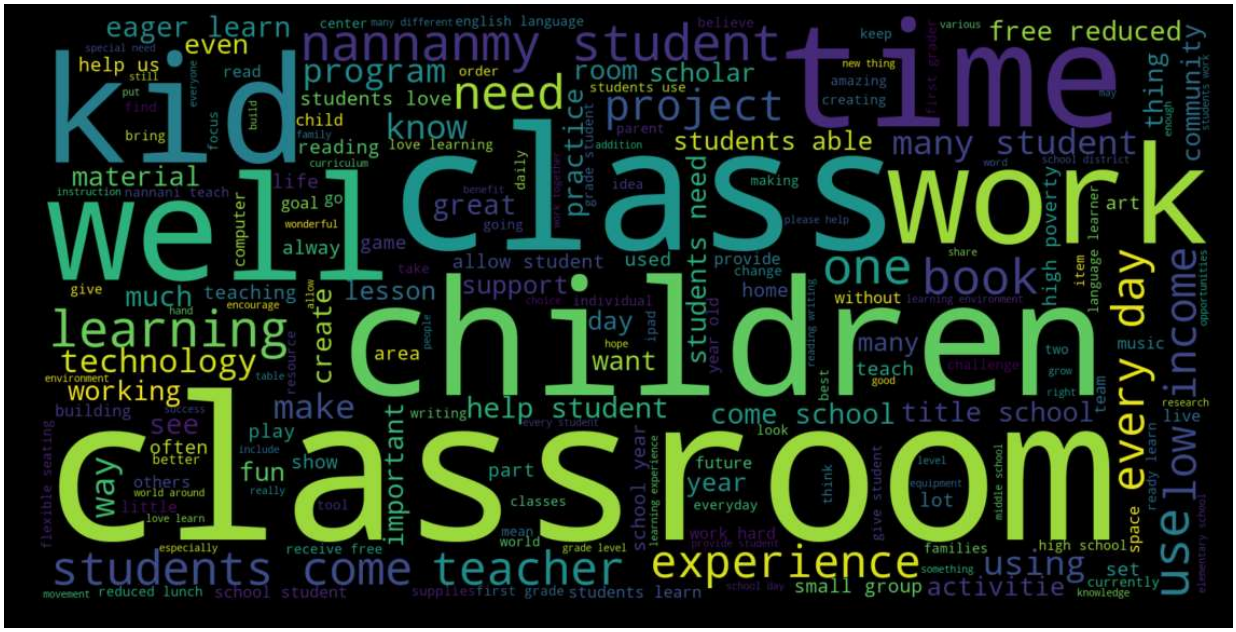
In [48]:
```
#cluster 4
words=''
for i in cluster4:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [49]:
```
#cluster 5
words=''
for i in cluster5:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [50]:
```
#cluster 6
words=''
for i in cluster6:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [ ]:
```
```

2.6 Apply AgglomerativeClustering

2.6.1 Dimensionality Reduction on the selected features

In [42]:
```
# using selectKBest we are find top 5k features


from sklearn.feature_selection import SelectKBest, chi2
t = SelectKBest(chi2,k=5000).fit(X_bow,y)
X_new = t.transform(X_bow)

print("="*100)

print("Final Data matrix on Bag of words")
print(X_new.shape)

print("="*100)
```

```
====================================================================================================
Final Data matrix on Bag of words
(50000, 5000)
====================================================================================================
```

In [43]:
```
X_tr = X_new[:5000]
X = X[:5000]
```

In [ ]:
```

```

For K=2

In [44]:
```
from sklearn.cluster import AgglomerativeClustering
aggcl=AgglomerativeClustering(n_clusters=2).fit(X_tr.toarray())
```

In [47]:
```
cluster1=[]
cluster2=[]
essays = X['essay'].values
for i in range(aggcl.labels_.shape[0]):
    if aggcl.labels_[i] == 0:
        cluster1.append(essays[i])
    elif aggcl.labels_[i] == 1:
        cluster2.append(essays[i])
```

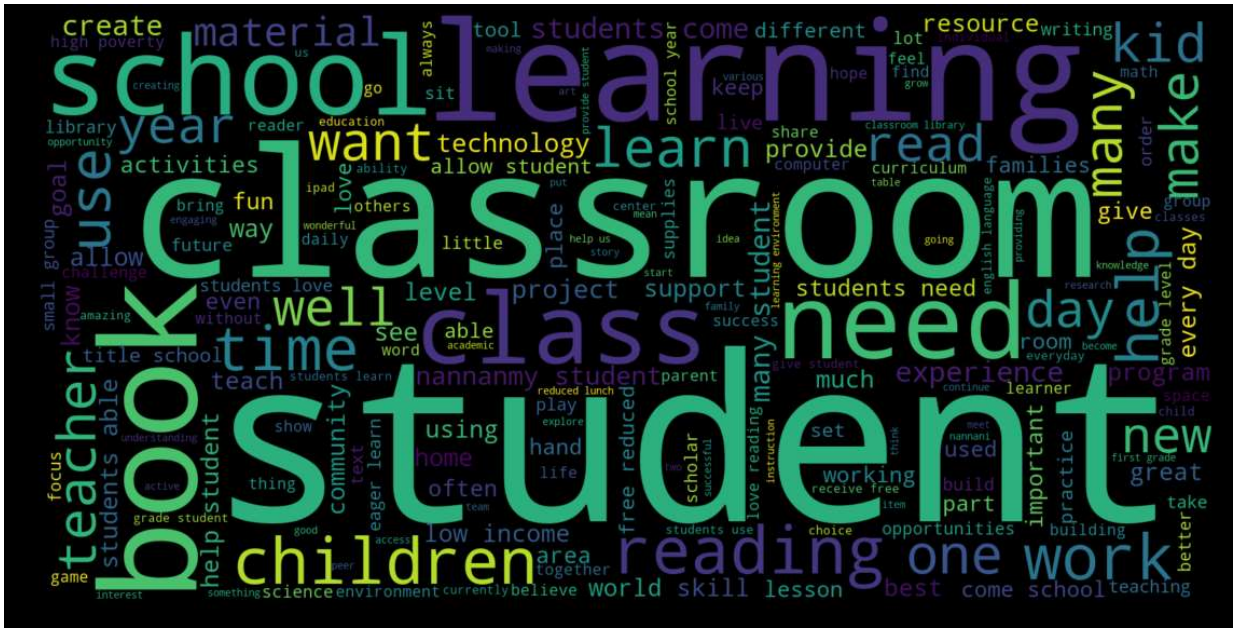
In [48]:
```
for i in range(3):
    print('%s\n'%(cluster1[i]))
```

```
my students english learners working english second third languages we melting
pot refugees immigrants native born americans bringing gift language school we
24 languages represented english learner program students every level mastery w
e also 40 countries represented families within school each student brings weal
th knowledge experiences us open eyes new cultures beliefs respect the limits l
anguage limits world ludwig wittgenstein our english learner strong support sys
tem home begs resources many times parents learning read speak english along si
de children sometimes creates barriers parents able help child learn phonetics
letter recognition reading skills by providing dvd players students able contin
ue mastery english language even no one home able assist all families students
within level 1 proficiency status offered part program these educational videos
specially chosen english learner teacher sent home regularly watch the videos h
elp child develop early reading skills parents not access dvd player opportunit
y check dvd player use year the plan use videos educational dvd years come el s
tudents nannan

our students arrive school eager learn they polite generous strive best they kn
ow education succeed life help improve lives our school focuses families low in
comes tries give student education deserve while not much students use material
s given best the projector need school crucial academic improvement students as
technology continues grow many resources internet teachers use growth students
however school limited resources particularly technology without disadvantage o
ne things could really help classrooms projector with projector not crucial ins
truction also growth students with projector show presentations documentaries p
hotos historical land sites math problems much with projector make teaching lea
rning easier also targeting different types learners classrooms auditory visual
kinesthetic etc nannan

i work unique school filled esl english second language high poverty students o
ur students individual personal struggles would break heart step doors would no
t notice anything positive resilient attitude learning my students love support
every aspect learning journey we students all world speaking 77 different langu
ages like family every student welcomed open arms regardless come language spea
k education love universal language classroom plenty my students live high pove
rty conditions limited no access technology ipads provide opportunity learn pla
ying reading math games engage inspire these ipads surely hottest commodity cla
ssroom students beg get play educational games i tracking data ipad several edu
cational programs help differentiate learning student students also able track
progress programs i look forward seeing students grow use ipads nannan
```

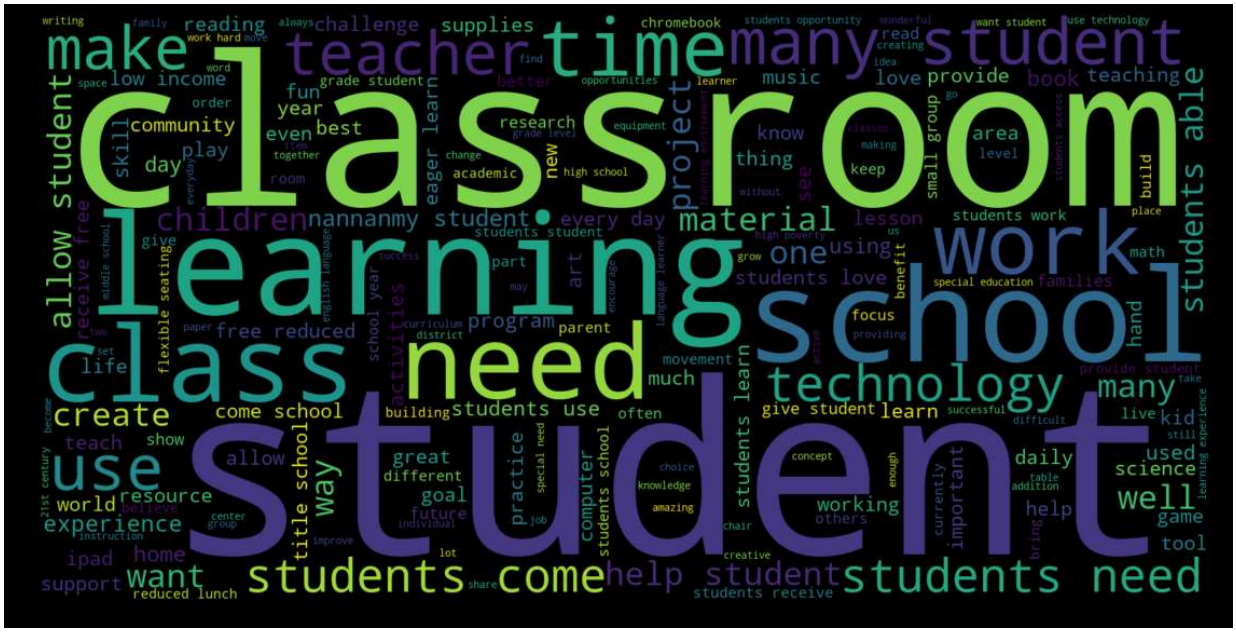
In [49]:
```
for i in range(3):
    print('%s\n'%(cluster2[i]))
```

```
true champions not always ones win guts by mia hamm this quote best describes s
tudents cholla middle school approach playing sports especially girls boys socc
er teams the teams made 7th 8th grade students not opportunity play organized s
port due family financial difficulties i teach title one middle school urban ne
ighborhood 74 students qualify free reduced lunch many come activity sport oppo
rtunity poor homes my students love participate sports learn new skills apart t
eam atmosphere my school lacks funding meet students needs i concerned lack exp
osure not prepare participating sports teams high school by end school year goa
l provide students opportunity learn variety soccer skills positive qualities p
erson actively participates team the students campus come school knowing face u
phill battle comes participating organized sports the players would thrive fiel
d confidence appropriate soccer equipment play soccer best abilities the studen
ts experience helpful person part team teaches positive supportive encouraging
others my students using soccer equipment practice games daily basis learn prac
tice necessary skills develop strong soccer team this experience create opportu
nity students learn part team positive contribution teammates the students get
opportunity learn practice variety soccer skills use skills game access type ex
perience nearly impossible without soccer equipment students players utilize pr
actice games nannan

i moving 2nd grade 3rd grade beginning next school year i takings current stude
nts move i teach inclusion classroom includes students adhd sld well autistic s
tudents my students work hard achieving goals no matter struggles may the schoo
l i teach houses great deal autistic students well ell students my student love
read work challenge they also love move around they work better able move room
different areas rather usual set these flexible seating options allow students
different seating options instead sitting traditional desk chair able use flexi
ble seating tools reduce stress anxiety these tools beneficial students special
needs also students it proven fact students moving oxygen going brain means lea
rning taking place these flexible seating options allow students move tradition
al seat allows reduce stress classroom this project significantly help students
reduce stress anxiety standardized testing the students 3rd grade required take
state mandated test this puts great deal stress students perform well test if s
tudents able work throughout year less stressful classroom assistance flexible
seating obtain skills needed successful standardized test nannan

not students struggle poverty also learning master english language minority st
udents represent 35 student population regardless background socioeconomic stat
us students deserve high quality education these children future these students
eager learn filled excitement opportunity use technology classroom however almo
st 650 students attending school 4 ipad carts entire building not get much expo
sure need as educator vital i try help become apart 21 century digital age thes
e ipads allow my students need 4 ipads latest technology classroom a long time
ago used paper pencils teaching students that time passed in 21st century stude
nts need latest technology stay ahead my students fully engaged use technology
hear pin drop room ipads make simplest tasks fun for example instead worksheets
use interactive apps practice math skills the mobility ipad also important stud
ents using around classroom sometimes areas school i requesting 4 ipads use cla
ssroom my students use listen digital books assignments study island study jams
interactive sites please help fund project when get opportunity use ipad cart s
chool i noticed beneficial student learning not donating project help students
also help future nannan
```

In [50]:
```
#cluster 1
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [51]:
```
#cluster 1
words=''
for i in cluster2:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

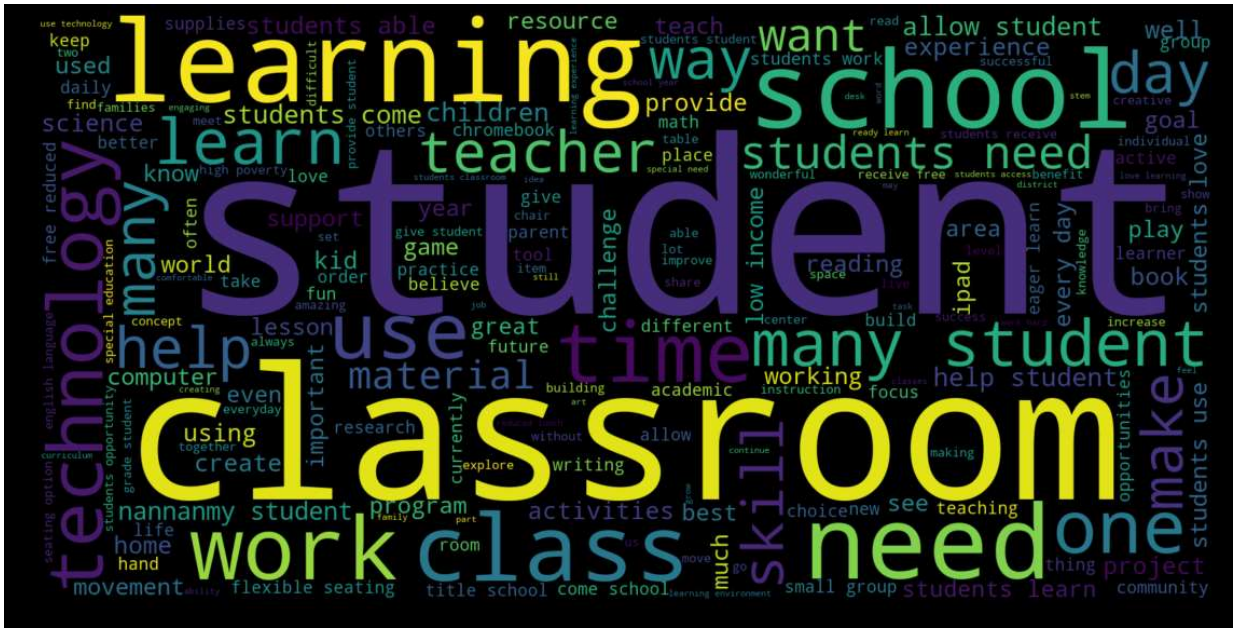


For K=5

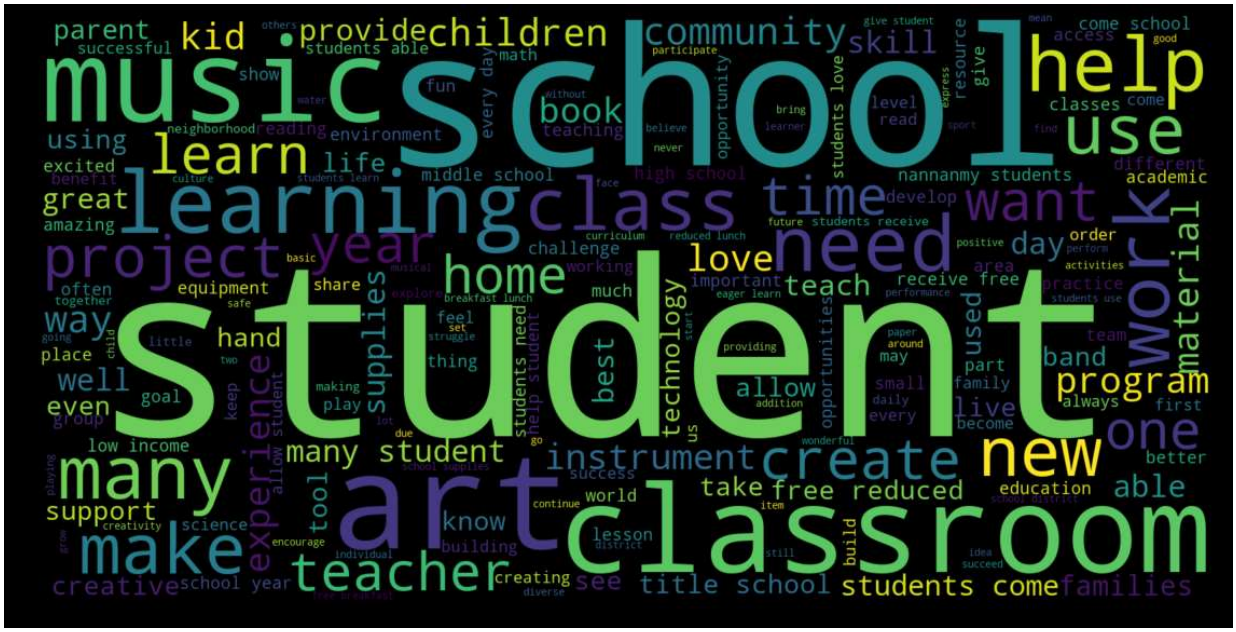
In [53]:
```
from sklearn.cluster import AgglomerativeClustering
aggcl=AgglomerativeClustering(n_clusters=5).fit(X_tr.toarray())
```

In [54]:
```
cluster1=[]
cluster2=[]
cluster3=[]
cluster4=[]
cluster5=[]
essays = X['essay'].values
for i in range(aggcl.labels_.shape[0]):
    if aggcl.labels_[i] == 0:
        cluster1.append(essays[i])
    elif aggcl.labels_[i] == 1:
        cluster2.append(essays[i])
    elif aggcl.labels_[i] == 2:
        cluster3.append(essays[i])
    elif aggcl.labels_[i] == 3:
        cluster4.append(essays[i])
    elif aggcl.labels_[i] == 4:
        cluster5.append(essays[i])
```

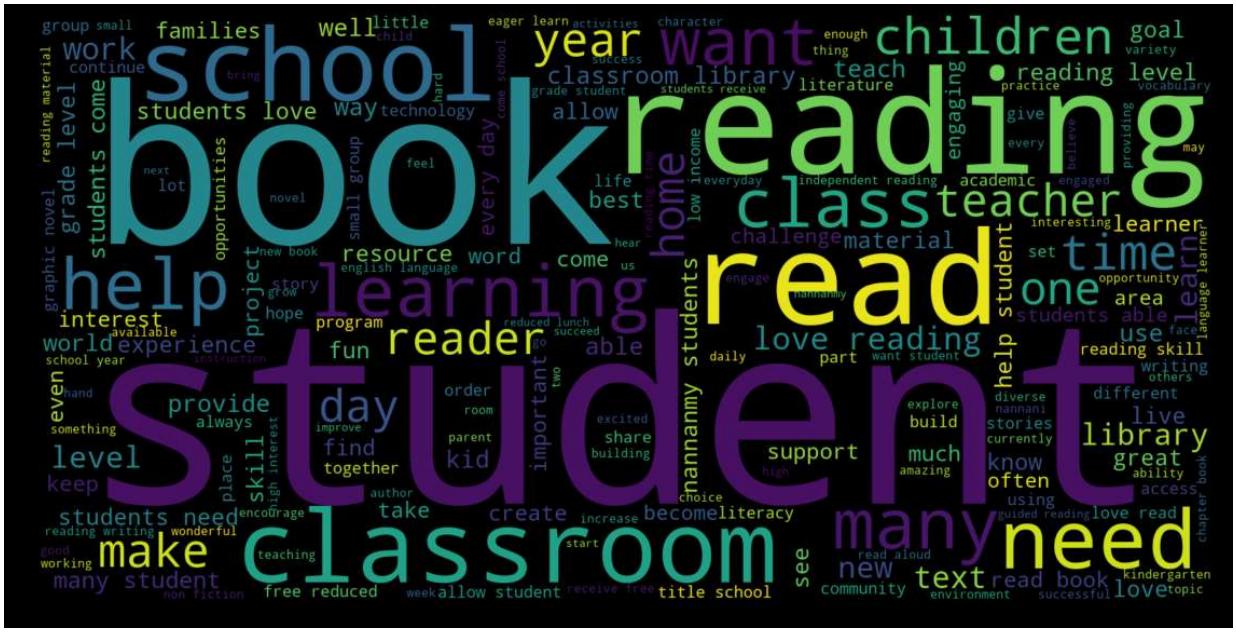
In [55]:
```
#cluster 1
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



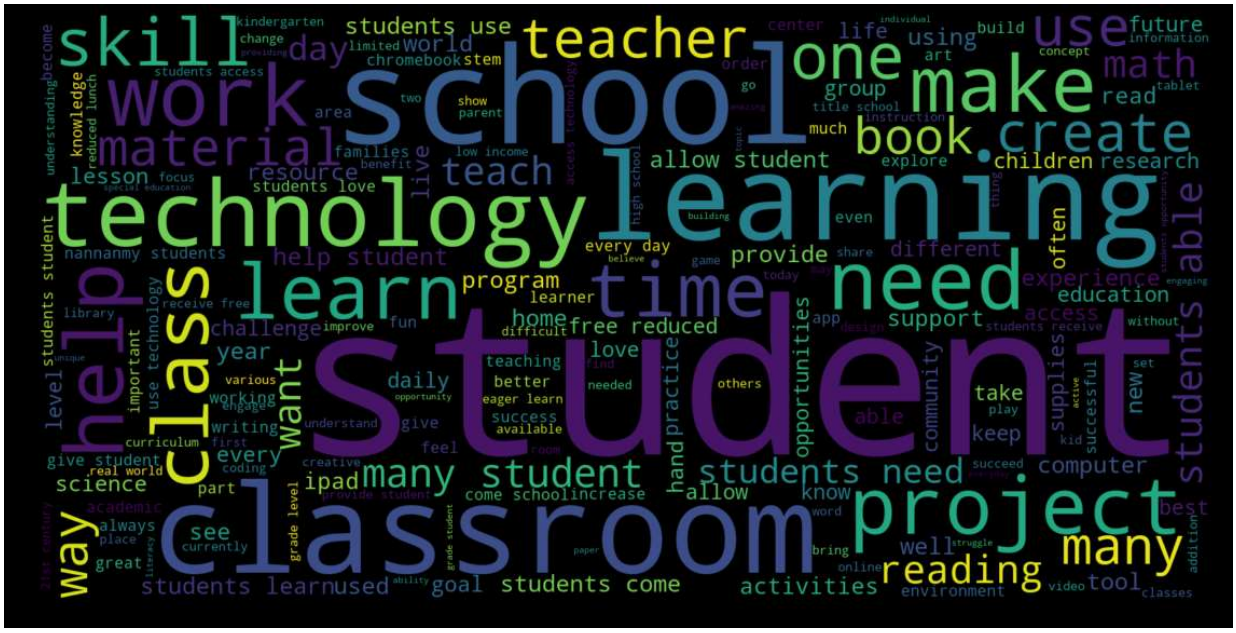
In [56]:
```
words=''
for i in cluster2:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



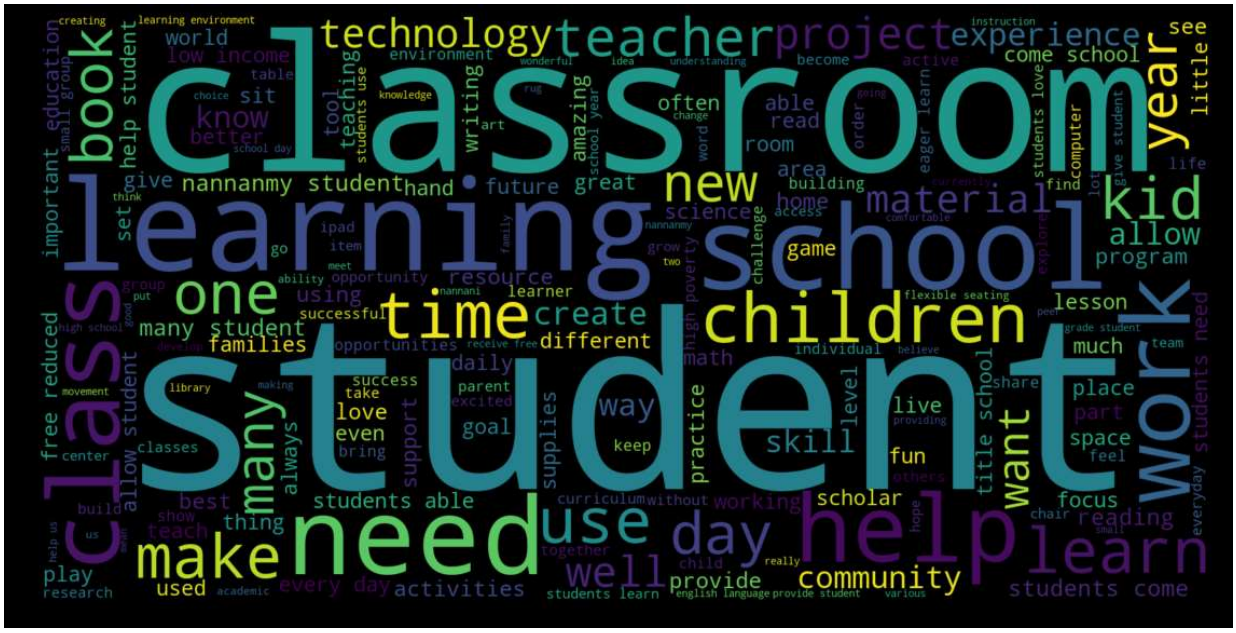
In [57]:
```
words=''
for i in cluster3:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [58]:
```
words=''
for i in cluster4:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [59]:
```
words=''
for i in cluster5:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



2.7 Apply DBSCAN

In [68]:
```
# using selectKBest we are find top 5k features


from sklearn.feature_selection import SelectKBest, chi2
t = SelectKBest(chi2,k=5000).fit(X_bow,y)
X_new = t.transform(X_bow)

print("="*100)

print("Final Data matrix on Bag of words")
print(X_new.shape)

print("="*100)
```

```
====================================================================================================
Final Data matrix on Bag of words
(50000, 5000)
====================================================================================================
```

In [69]:
```
X_tr = X_new[:5000]
X = X[:5000]
```

In [70]:
```
from sklearn.preprocessing import StandardScaler
# dat=StandardScaler().fit_transform(X_tr.toarray())
dat=X_tr.toarray()
dat
```

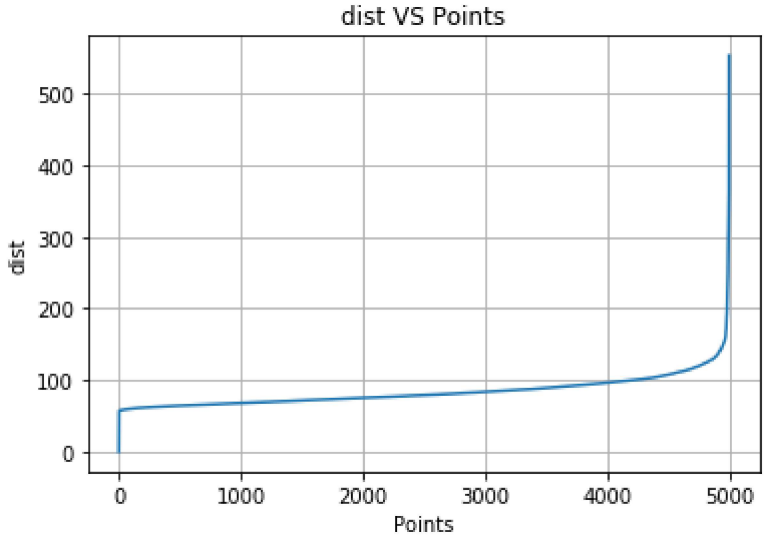
Out[70]:
```
array([[0., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       ...,
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.]])
```

In [63]:
```
from sklearn.metrics.pairwise import euclidean_distances
euclidean_distances(dat, dat[464].reshape(1, -1))
```

Out[63]:
```
array([[26.26785107],
       [23.51595203],
       [31.41655614],
       ...,
       [23.        ],
       [31.51190251],
       [24.2693222 ]])
```

In [72]:
```
min_points = 1500
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import euclidean_distances
datt=StandardScaler().fit_transform(dat)
distance=[]
for point in tqdm(datt):
    temp = euclidean_distances(datt, point.reshape(1, -1))
    distance.append(temp[min_points])
sorted_distance = np.sort(np.array(distance))


sorted_dist = np.sort(sorted_distance.reshape(1,-1)[0])
points = [i for i in range(len(datt))]
# Draw distances(d_i) VS points(x_i) plot
plt.plot(points, sorted_dist)
plt.xlabel('Points')
plt.ylabel('dist')
plt.title('dist VS Points')
plt.grid()
plt.show()
```

```
100%|██████████| 5000/5000 [12:55<00:00,  7.58it/s]
```

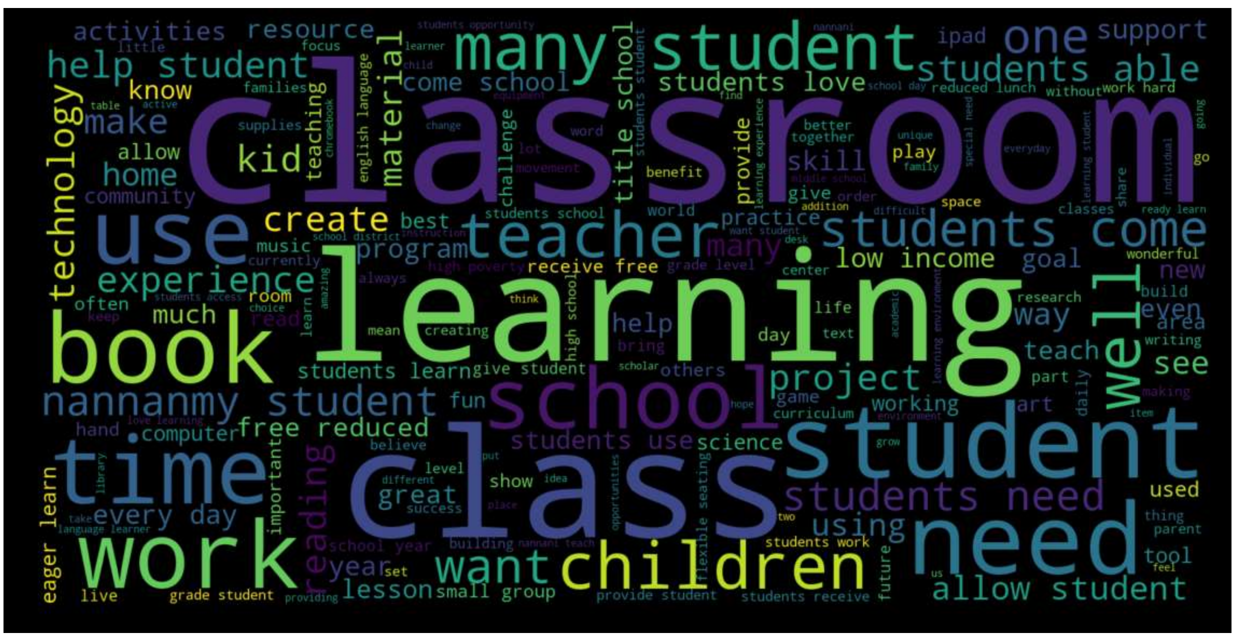


In [73]:
```
#we can see that point of inflexion is at eps=9
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=90,n_jobs=-1)
dbscan.fit(datt)
print('No of clusters: ',len(set(dbscan.labels_)))
print('Cluster are including noise i.e -1: ',set(dbscan.labels_))
```

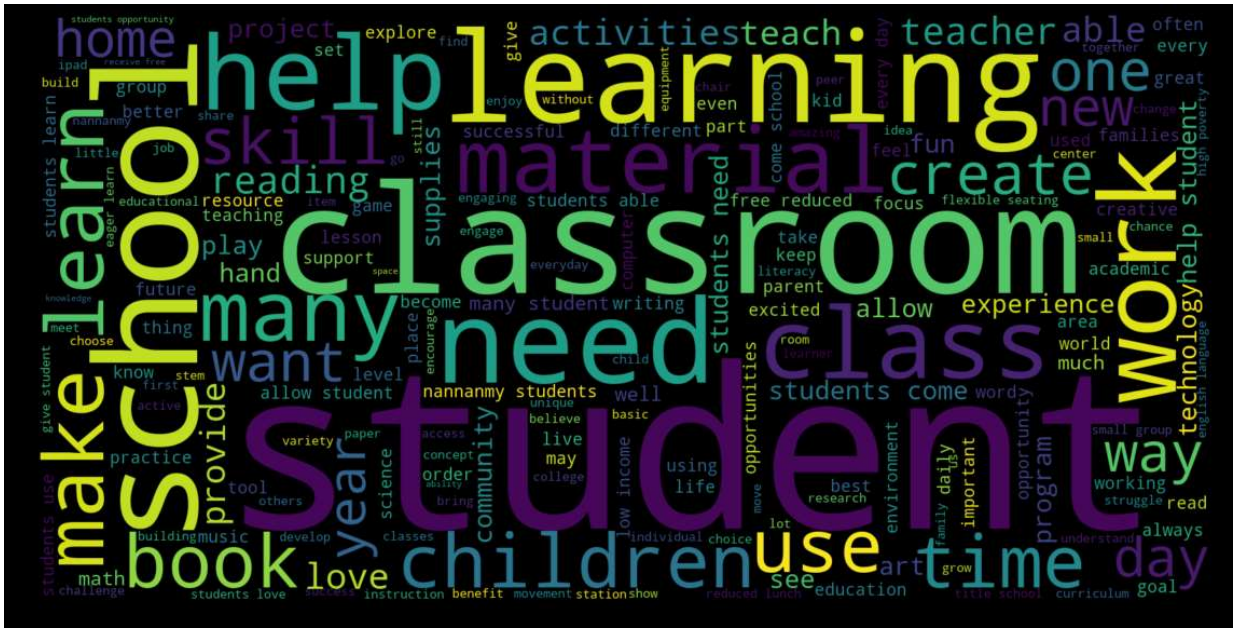
```
No of clusters:  2
Cluster are including noise i.e -1:  {0, -1}
```

In [74]:
```
#ignoring -1 as it is for noise
cluster1=[]
noisecluster1=[]
for i in range(dbscan.labels_.shape[0]):
    if dbscan.labels_[i] == 0:
        cluster1.append(essays[i])
    elif dbscan.labels_[i] == -1:
        noisecluster1.append(essays[i])
```

In [75]:
```
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



In [76]:
```
words=''
for i in noisecluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


# Display the generated image:
plt.figure( figsize=(20,10), facecolor='k')
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```



3. Conclusions

a. K-Means:

1.First we have done hyperparameter tuning of K_values with 2,3,4,5,6,7,8 and got the inflection point at k=6.

2.There after, we trained K-Means on K_value=6.

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

4.And finally we plot the word cloud.

b. Agglomerative:

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

2.Then we applied Agglomerative clustering on k=2.

3.Then clustered the essays into 2 seperate clusters.

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

5.Then applied Agglomerative clustering on k=5.

6.Then we clustered the essays into 5 seperate clusters.

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

c. DBScan:

1. Firstly we converted the reduced sparce matrix to dense using toarray().
2. Then we transformed the data to standard scalar.
3. Then we computed euclidean distance for every point to every other point and took the distance of min_pts.
4. Obtained the best eps to be 90 from the above graph b/w dist and points.
5. Then formed clusters on noise points and non-noise points.
6. Printed the essays in each of the two clusters.
7. Then printed the wordcloud.

In [ ]: