Donors_Choose_K-Means_Agglomerative_DBSCAN_Assignment

In [60]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import sqlite3
import pandas as pd
import numpy as np
import nltk
import gc
gc.enable()
gc.DEBUG_SAVEALL
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

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

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

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

from tqdm import tqdm
import os
import math
from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
gc.set_threshold(2, 1, 1)
```

2.1 Loading Input Data

In [61]:
```
# %load_ext memory_profiler


# s=0
# We are taking samples of 0's and 1's and appending them to overcome memory erro


project_data = pd.read_csv('train_data.csv')
# project_data=project_data.dropna(how='any')
# project_data_1 = project_data[project_data['project_is_approved'] == s+1]
# project_data_0 = project_data[project_data['project_is_approved'] == s]
project_data=project_data.fillna("")
project_data_1=project_data.head(6000)
project_data_0=project_data.tail(6000)
project_data_1=project_data_1.append(project_data_0)
project_data=project_data_1
resource_data = pd.read_csv('resources.csv')

#Sorting them by columns to spread the zeros and one's unevenly in the 'project_
# project_data.sort_values(by=['project_essay_1'])
# project_data.sort_values(by=['project_essay_2'], ascending=False)
# project_data.sort_values(by=['project_essay_3'])
# project_data.sort_values(by=['project_essay_4'], ascending=False)
project_data_1=None
project_data_0=None
```

In [62]:
```
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 (12000, 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 [63]:
```
print("Number of data points in resource data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(1)
# project_data.head(2)
```

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

Out[63]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.0

In [64]:
```
y = project_data['project_is_approved'].values
X = project_data.drop(['project_is_approved'], axis=1)
X.head(1)
project_data=None
gc.collect()
gc.enable()
gc.DEBUG_SAVEALL
```

Out[64]: 32

2.2 Getting the Data Model Ready:Preprocessing and Vectorizing categorical features

2.2.1 Preprocessing:project_grade_category

In [65]:
```
sub_catogories = list(X['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-

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())
X['project_grade_category'] = sub_cat_list
```

In [66]:
```
sub_catogories=None
sub_cat_list=None
temp=None
i=None
j=None
catogories=None
cat_list=None
temp=None
my_counter=None
word=None
cat_dict=None
gc.collect()
gc.enable()
gc.DEBUG_SAVEALL
```

Out[66]: 32

2.2.2 Preprocessing:project_subject_categories

In [67]:
```
catogories = list(X['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())

X['clean_categories'] = cat_list
X.drop(['project_subject_categories'], axis=1, inplace=True)
X.head(2)
```

Out[67]:

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

2.2.3 Preprocessing:project_subject_subcategories

In [68]:
```
sub_catogories = list(X['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())
```

In [69]:
```
X['clean_subcategories'] = sub_cat_list
X.drop(['project_subject_subcategories'], axis=1, inplace=True)
X.head(2)
```

Out[69]:

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

2.2.4 New Column:digits in summary

In [70]:
```
# Creating a new column 'digits_in_summary' which contains flags of 1 for /
# 'project_resource_summary' containing numeric specification in their requiremn
project_resource_summary = []
new=[]

project_resource_summary = list(X['project_resource_summary'].values)

for i in project_resource_summary:
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(' '):
        if j.isdigit():
            new.append(1)
            break
        else:
            continue
    else:
        new.append(0)


X['digits_in_summary']=new
```

In [71]:
```
#To make best use of the memory we are setting the variable names to 'None' and p
project_resource_summary=None
new=None
new1=None
i=None
j=None
a=None

gc.collect()
gc.enable()
gc.DEBUG_SAVEALL
```

Out[71]: 32

2.2.5 Preprocessing:Text features (Project Essay's)

In [72]:
```
# merge two column text dataframe:
X["essay"] = X["project_essay_1"].map(str) +\
                        X["project_essay_2"].map(str) + \
                        X["project_essay_3"].map(str) + \
                        X["project_essay_4"].map(str)
```

In [73]:
```
X = X.drop(['project_essay_1', 'project_essay_2','project_essay_3', 'project_essa
X.shape
```

Out[73]: (12000, 14)

2.2.6 Adding column Cost per project in dataset

In [74]:
```
# 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)
type(price_data)
```

Out[74]:
```
pandas.core.frame.DataFrame
```

In [75]:
```
# join two dataframes in python:
X = pd.merge(X, price_data, on='id', how='left')
X.head(2)
```

Out[75]:

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

In [76]:
```
#To make best use of the memory we are setting the variable names to 'None' and p
resource_data=None
price_data=None
gc.collect()
gc.enable()
gc.DEBUG_SAVEALL
```

Out[76]:
```
32
```

2.2.7 Text Preprocessing:Essay Text

In [77]:
```
# 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 [78]:
```
sent = decontracted(X['essay'].values[99])
print(sent)
print("="*50)
```

```
My preschool students are children who are three to five years of age.  My scho
ol is in sunny San Pedro, California. The children from San Pedro come to schoo
l each morning ready to learn and grow.  There is never a dull moment in our cl
ass; my students are busy bees moving from one interest area to another.  They
are eager to learn, explore, and experiment with the instructional materials an
d centers I set up for them.  We need more materials for the children to engage
with, materials that will foster their interest in technology, literacy, math,
science, art, and engineering. \r\nMy student is will learn number recognition
and develop counting skills while engaging with the Learn to count picture puzz
les and number Sequencing puzzles. While building with the 3-D Magnet Builders
and Crystal Building Blocks, my student is mathematical skills will be supporte
d and strengthened in concepts such as measurement, comparison, number estimati
on, symmetry and balance. My student is will build number skills as the they si
ft and make exciting number shell discoveries with every scoop at the sand tabl
e. The sort a shape activity board will allow my youngest students to learn col
ors, shapes and sorting skills as they fit various shape pieces into place.
==================================================
```

In [79]:
```
# \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)
```

```
My preschool students are children who are three to five years of age.  My scho
ol is in sunny San Pedro, California. The children from San Pedro come to schoo
l each morning ready to learn and grow.  There is never a dull moment in our cl
ass; my students are busy bees moving from one interest area to another.  They
are eager to learn, explore, and experiment with the instructional materials an
d centers I set up for them.  We need more materials for the children to engage
with, materials that will foster their interest in technology, literacy, math,
science, art, and engineering.   My student is will learn number recognition an
d develop counting skills while engaging with the Learn to count picture puzzle
s and number Sequencing puzzles. While building with the 3-D Magnet Builders an
d Crystal Building Blocks, my student is mathematical skills will be supported
and strengthened in concepts such as measurement, comparison, number estimatio
n, symmetry and balance. My student is will build number skills as the they sif
t and make exciting number shell discoveries with every scoop at the sand tabl
e. The sort a shape activity board will allow my youngest students to learn col
ors, shapes and sorting skills as they fit various shape pieces into place.
```

In [80]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My preschool students are children who are three to five years of age My school
is in sunny San Pedro California The children from San Pedro come to school eac
h morning ready to learn and grow There is never a dull moment in our class my
students are busy bees moving from one interest area to another They are eager
to learn explore and experiment with the instructional materials and centers I
set up for them We need more materials for the children to engage with material
s that will foster their interest in technology literacy math science art and e
ngineering My student is will learn number recognition and develop counting ski
lls while engaging with the Learn to count picture puzzles and number Sequencin
g puzzles While building with the 3 D Magnet Builders and Crystal Building Bloc
ks my student is mathematical skills will be supported and strengthened in conc
epts such as measurement comparison number estimation symmetry and balance My s
tudent is will build number skills as the they sift and make exciting number sh
ell discoveries with every scoop at the sand table The sort a shape activity bo
ard will allow my youngest students to learn colors shapes and sorting skills a
s they fit various shape pieces into place
```

In [81]:
```
# 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 [82]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(X['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%|████████████████████████████████████| 12000/12000 [00:20<00:00, 573.99it/s]
```

In [85]:
```
# after preprocesing

# X['essay'] = None
X['essay'] = preprocessed_essays

X.head(2)
```

Out[85]:

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

In [25]:
```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_project_title = []
# tqdm is for printing the status bar
for sentance in tqdm(X['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_project_title.append(sent.lower().strip())
```

100%|██████████| 12000/12000 [00:00<00:00, 15624.11it/s]

In [26]:
```
preprocessed_project_title[4999]
# after preprocesing

# X['project_title'] = None
X['project_title'] = preprocessed_project_title

X.head(2)
```

Out[26]:

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

2.2.8 Splitting the data into Train and Test

In [27]:
```
# # train test split(67:33)
# from sklearn.model_selection import train_test_split
# X, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y,
# # X, X_cv, y_train, y_cv = train_test_split(X, y_train, test_size=0.33, stratij
```

In [28]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in X['clean_categories'].values:
    my_counter.update(word.split())
print(my_counter)
# dict sort by value python: https://stackoverflow.com/a/613218/4084039
cat_dict = dict(my_counter)
sorted_cat_dict_train = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

```
Counter({'Literacy_Language': 5755, 'Math_Science': 4588, 'Health_Sports': 156
2, 'SpecialNeeds': 1517, 'AppliedLearning': 1365, 'Music_Arts': 1135, 'History_
Civics': 633, 'Warmth': 137, 'Care_Hunger': 137})
```

2.2.9 Vectorizing Categorical data: clean_categories(Project subject categories)

In [29]:
```
print(X.shape)
# print(X_cv.shape, y_cv.shape)
# print(X_test.shape, y_test.shape)

print("="*100)

Bow_features_names1=[]

from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict_train.keys()), lowe
vectorizer.fit(X['clean_categories'].values) # fit has to happen only on train d

# we use the fitted Countvectorizer to convert the text to vector
X_clean_cat_ohe = vectorizer.transform(X['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_clean_cat_ohe.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(vectorizer_test.get_feature_names())
print("="*100)
```

```
(12000, 16)
============================================================================
====================
After vectorizations
(12000, 9)
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning', 'S
pecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
============================================================================
====================
```

2.2.10 Vectorizing Categorical data: clean_subcategories(Project subject subcategories)

In [30]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in X['clean_subcategories'].values:
    my_counter.update(word.split())
sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict_train = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1
```

In [31]:
```
print(X.shape)
# print(X_cv.shape, y_cv.shape)
# print(X_test.shape, y_test.shape)

print("="*100)

vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict_train.keys()),
vectorizer.fit(X['clean_subcategories'].values) # fit has to happen only on trai

# we use the fitted Countvectorizer to convert the text to vector
X_clean_sub_cat_ohe = vectorizer.transform(X['clean_subcategories'].values)
# X_cv_clean_sub_cat_ohe = vectorizer.transform(X_cv['clean_subcategories'].value
# X_test_clean_sub_cat_ohe = vectorizer.transform(X_test['clean_subcategories'].

print("After vectorizations")
print(X_clean_sub_cat_ohe.shape)
# print(X_cv_clean_sub_cat_ohe.shape, y_cv.shape)
# print(X_test_clean_sub_cat_ohe.shape, y_test.shape)

print(vectorizer.get_feature_names())
# print(vectorizer_test.get_feature_names())
print("="*100)
```

```
(12000, 16)
============================================================================
====================
After vectorizations
(12000, 30)
['Economics', 'FinancialLiteracy', 'CommunityService', 'ParentInvolvement', 'Ci
vics_Government', 'Extracurricular', 'ForeignLanguages', 'NutritionEducation',
'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'CharacterEducatio
n', 'Other', 'TeamSports', 'College_CareerPrep', 'History_Geography', 'Music',
'ESL', 'EarlyDevelopment', 'Health_LifeScience', 'Gym_Fitness', 'EnvironmentalS
cience', 'VisualArts', 'Health_Wellness', 'AppliedSciences', 'SpecialNeeds', 'L
iterature_Writing', 'Mathematics', 'Literacy']
============================================================================
====================
```

2.2.11 Vectorizing Categorical data: school_state

In [32]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in X['school_state'].values:
    my_counter.update(word.split())
school_state_dict = dict(my_counter)
sorted_school_state_dict_train = dict(sorted(school_state_dict.items(), key=lambd
# sorted_school_state_dict
```

In [33]:
```
print(X.shape)
# print(X_cv.shape, y_cv.shape)
# print(X_test.shape, y_test.shape)

print("="*100)


vectorizer = CountVectorizer(vocabulary=list(sorted_school_state_dict_train.keys
vectorizer.fit(X['school_state'].values) # fit has to happen only on train data

# we use the fitted Countvectorizer to convert the text to vector
X_School_state_ohe = vectorizer.transform(X['school_state'].values)
# X_cv_School_state_ohe = vectorizer.transform(X_cv['school_state'].values)
# X_test_School_state_ohe = vectorizer.transform(X_test['school_state'].values)

print("After vectorizations")
print(X_School_state_ohe.shape)
# print(X_cv_School_state_ohe.shape, y_cv.shape)
# print(X_test_School_state_ohe.shape, y_test.shape)

print(vectorizer.get_feature_names())
# print(vectorizer_test.get_feature_names())
print("="*100)
```

```
(12000, 16)
====================================================================================
====================
After vectorizations
(12000, 51)
['VT', 'WY', 'ND', 'MT', 'NE', 'DE', 'RI', 'AK', 'NH', 'SD', 'DC', 'WV', 'ME',
'HI', 'NM', 'KS', 'IA', 'ID', 'AR', 'CO', 'MN', 'MS', 'KY', 'OR', 'NV', 'MD',
'UT', 'WI', 'AL', 'TN', 'CT', 'VA', 'AZ', 'WA', 'NJ', 'OK', 'IN', 'MA', 'MO',
'OH', 'LA', 'PA', 'MI', 'GA', 'SC', 'IL', 'NC', 'FL', 'TX', 'NY', 'CA']
====================================================================================
====================
```

2.2.12 Vectorizing Categorical data: project_grade_category

In [34]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
for word in X['project_grade_category'].values:
    my_counter.update(word.split())
project_grade_dict = dict(my_counter)
sorted_project_grade_dict_train = dict(sorted(project_grade_dict.items(), key=la
```

In [35]:
```
print(X.shape)
# print(X_cv.shape, y_cv.shape)
# print(X_test.shape, y_test.shape)

print("="*100)

vectorizer= CountVectorizer(vocabulary=list(sorted_project_grade_dict_train.keys
vectorizer.fit(X['project_grade_category'].values) # fit has to happen only on t

# we use the fitted Countvectorizer_pro_gradeto convert the text to vector
X_project_grade_category_ohe = vectorizer.transform(X['project_grade_category'].
# X_cv_project_grade_category_ohe = vectorizer.transform(X_cv['project_grade_cat
# X_test_project_grade_category_ohe = vectorizer.transform(X_test['project_grade_


print("After vectorizations")
print(X_project_grade_category_ohe.shape)
# print(X_cv_project_grade_category_ohe.shape, y_cv.shape)
# print(X_test_project_grade_category_ohe.shape, y_test.shape)

print(vectorizer.get_feature_names())
# print(vectorizer_test.get_feature_names())
print("="*100)
```

```
(12000, 16)
====================================================================================
====================
After vectorizations
(12000, 4)
['Grades9-12', 'Grades6-8', 'Grades3-5', 'GradesPreK-2']
====================================================================================
====================
```

2.2.13 Vectorizing Categorical data: teacher_prefix

In [36]:
```
#To overcome the blanks in the teacher_prefix categry the .fillna is used
X['teacher_prefix']=X['teacher_prefix'].fillna("")
# project_data1=project_data.dropna()
```

In [37]:
```
#To overcome the blanks in the teacher_prefix categry the .fillna is used
# X_test['teacher_prefix']=X_test['teacher_prefix'].fillna("")
# project_data1=project_data.dropna()
```

In [38]:
```
# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4
from collections import Counter
my_counter = Counter()
my_counter1=[]
# project_data['teacher_prefix']=str(project_data['teacher_prefix'])
for word in X['teacher_prefix'].values:
    my_counter.update(word.split())
teacher_prefix_dict = dict(my_counter)
sorted_teacher_prefix_dict_train = dict(sorted(teacher_prefix_dict.items(), key=l
# teacher_prefix_dict
```

In [39]:
```
print(X.shape)
# print(X_cv.shape, y_cv.shape)
# print(X_test.shape, y_test.shape)

print("="*100)

vectorizer = CountVectorizer(vocabulary=list(sorted_teacher_prefix_dict_train.key
vectorizer.fit(X['teacher_prefix'].values) # fit has to happen only on train data

# we use the fitted Countvectorizer to convert the text to vector
X_teacher_prefix_ohe = vectorizer.transform(X['teacher_prefix'].values)
# X_cv_teacher_prefix_ohe = vectorizer.transform(X_cv['teacher_prefix'].values)
# X_test_teacher_prefix_ohe = vectorizer.transform(X_test['teacher_prefix'].value

print("After vectorizations")
print(X_teacher_prefix_ohe.shape)
# print(X_cv_teacher_prefix_ohe.shape, y_cv.shape)
# print(X_test_teacher_prefix_ohe.shape, y_test.shape)

print(vectorizer.get_feature_names())
# print(vectorizer_test.get_feature_names())
print("="*100)
```

```
(12000, 16)
====================================================================================
====================
After vectorizations
(12000, 4)
['Teacher', 'Mr.', 'Ms.', 'Mrs.']
====================================================================================
====================
```

2.3 Make Data Model Ready: Vectorizing Numerical features

2.3.1 Vectorizing Numerical features--Price

In [40]:
```
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import StandardScaler
normalizer = StandardScaler() #Normalizer()
# normalizer_test = Normalizer()
# normalizer.fit(X['price'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X['price'].values.reshape(-1,1))
# normalizer_test.fit(X_test['price'].values.reshape(1,-1))

X_price_norm = normalizer.transform(X['price'].values.reshape(-1,1))
# X_cv_price_norm = normalizer.transform(X_cv['price'].values.reshape(-1,1))
# X_test_price_norm = normalizer.transform(X_test['price'].values.reshape(-1,1))


# X_price_norm=np.reshape(X_price_norm,(1,-1))
# X_test_price_norm=np.reshape(X_test_price_norm,(1,-1))

print("After vectorizations")

# np.reshape(X_price_norm,
print(X_price_norm.shape)
# print(X_cv_price_norm.shape, y_cv.shape)
# print(X_test_price_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(12000, 1)
====================================================================================================
```

2.3.2 Vectorizing Numerical features--teacher_number_of_previously_posted_projects

In [41]:
```
from sklearn.preprocessing import Normalizer
normalizer_train = StandardScaler() #Normalizer()
# normalizer_test = StandardScaler() #Normalizer()
# normalizer.fit(X['teacher_number_of_previously_posted_projects'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X['teacher_number_of_previously_posted_projects'].values.reshape(

X_teacher_number_of_previously_posted_projects_norm = normalizer.transform(X['te
# X_cv_teacher_number_of_previously_posted_projects_norm = normalizer.transform(
# X_test_teacher_number_of_previously_posted_projects_norm = normalizer.transfor



# X_teacher_number_of_previously_posted_projects_norm=np.reshape(X_teacher_numbe
# X_test_teacher_number_of_previously_posted_projects_norm=np.reshape(X_test_tea


print("After vectorizations")
print(X_teacher_number_of_previously_posted_projects_norm.shape)
# print(X_cv_teacher_number_of_previously_posted_projects_norm.shape, y_cv.shape,
# print(X_test_teacher_number_of_previously_posted_projects_norm.shape, y_test.sh
print("="*100)
```

```
After vectorizations
(12000, 1)
====================================================================================================
```

2.3.3 Vectorizing Numerical features--digits_in_summary

In [42]:
```
X['digits_in_summary'].fillna(X['digits_in_summary'].mean(), inplace=True)
# X_cv['digits_in_summary'].fillna(X_cv['digits_in_summary'].mean(), inplace=True
# X_test['digits_in_summary'].fillna(X_test['digits_in_summary'].mean(), inplace=
```

In [43]:
```
from sklearn.preprocessing import Normalizer
normalizer_train = StandardScaler() #Normalizer()
# normalizer_test = StandardScaler() #Normalizer()
# normalizer.fit(X['digits_in_summary'].values)
# this will rise an error Expected 2D array, got 1D array instead:
# array=[105.22 215.96  96.01 ... 368.98  80.53 709.67].
# Reshape your data either using
# array.reshape(-1, 1) if your data has a single feature
# array.reshape(1, -1)  if it contains a single sample.
normalizer.fit(X['digits_in_summary'].values.reshape(-1,1))

X_digits_in_summary_norm = normalizer.transform(X['digits_in_summary'].values.re
# X_cv_digits_in_summary_norm = normalizer.transform(X_cv['digits_in_summary'].v
# X_test_digits_in_summary_norm = normalizer.transform(X_test['digits_in_summary


# X_digits_in_summary_norm=np.reshape(X_digits_in_summary_norm,(1,-1))
# X_test_digits_in_summary_norm=np.reshape(X_test_digits_in_summary_norm,(1,-1))

print("After vectorizations")
print(X_digits_in_summary_norm.shape)
# print(X_cv_digits_in_summary_norm.shape, y_cv.shape)
# print(X_test_digits_in_summary_norm.shape, y_test.shape)
print("="*100)
```

```
After vectorizations
(12000, 1)
====================================================================================
====================
```

2.4 Make Data Model Ready: Vectorizing Essay and Project_title feature into BOW

Vectorizing Text data

2.4.1 Bag of words:Essays

In [44]:
```
print(X.shape)
# print(X_cv.shape, y_cv.shape)
# print(X_test.shape, y_test.shape)

print("="*100)

# We are considering only the words which appeared in at least 10 documents(rows

vectorizer = CountVectorizer(min_df=10,ngram_range=(2,2), max_features=5000)
vectorizer.fit(X['essay'].values) # fit has to happen only on train data

# we use the fitted Countvectorizer to convert the text to vector
X_essay_bow = vectorizer.transform(X['essay'].values)
# X_cv_essay_bow = vectorizer.transform(X_cv['essay'].values)
# X_test_essay_bow = vectorizer.transform(X_test['essay'].values)

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

```
(12000, 16)
====================================================================================
====================
After vectorizations
(12000, 5000)
====================================================================================
====================
```

2.4.2 Bag of words:Project Title

In [45]:
```
print(X.shape)
# print(X_cv.shape, y_cv.shape)
# print(X_test.shape, y_test.shape)

print("="*100)


vectorizer = CountVectorizer(min_df=10,ngram_range=(2,2), max_features=5000)
vectorizer.fit(X['project_title'].values) # fit has to happen only on train data

# we use the fitted Countvectorizer to convert the text to vector
X_project_title_bow = vectorizer.transform(X['project_title'].values)
# X_cv_project_title_bow = vectorizer.transform(X_cv['project_title'].values)
# X_test_project_title_bow = vectorizer.transform(X_test['project_title'].values)

print("After vectorizations")
print(X_project_title_bow.shape)
# print(X_cv_project_title_bow.shape, y_cv.shape)
# print(X_test_project_title_bow.shape, y_test.shape)

print("="*100)
```

```
(12000, 16)
====================================================================================================
After vectorizations
(12000, 263)
====================================================================================================
```

In [46]:
```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense
X_BOW = hstack((X_digits_in_summary_norm,X_teacher_number_of_previously_posted_pr
X_BOW=X_BOW.todense()
X_BOW=np.array(X_BOW)

# X_BOW_cv = hstack((X_cv_project_title_bow,X_cv_essay_bow ,X_cv_digits_in_summar
# X_BOW_cv=X_BOW_cv.todense()
# X_BOW_cv=np.array(X_BOW_cv)

# X_BOW_test = hstack((X_test_digits_in_summary_norm,X_test_teacher_number_of_pre
# X_BOW_test=X_BOW_test.todense()
# X_BOW_test=np.array(X_BOW_test)

X_project_title_bow=None
X_essay_bow =None

# X_test_project_title_bow=None
# X_test_essay_bow =None
```

2.5 Best K

2.5.1 Selecting the best K features

In [47]:
```
# Selecting 2000 best features from Tfidf to see the variation in the AUC
from sklearn.feature_selection import SelectKBest, f_classif
X_BOW = SelectKBest(f_classif, k=5000).fit_transform(X_BOW,y)
X_BOW.shape
# X_BOW_test = SelectKBest(f_classif, k=5000).fit_transform(X_BOW_test,y_test)
# X_BOW_test.shape


# using selectKBest we are find top 5k features


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

# print("="*100)

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

# print("="*100)
```

```
C:\Users\Admin\Anaconda3\lib\site-packages\sklearn\feature_selection\_univariat
e_selection.py:114: UserWarning:

Features [5306 5307 5308 5309 5310 5311 5312] are constant.
```

Out[47]:
```
(12000, 5000)
```

In [87]:
```
X_trt=X_BOW[:11000]
X_BOW = X_BOW[:5000]
X = X[:11000]
X_BOW.shape
```

Out[87]:
```
(5000, 5000)
```

2.7 Applying DBSCAN on the data

In [49]:
```
from sklearn.metrics.pairwise import euclidean_distances
euclidean_distances(X_trt, X_trt[464].reshape(1, -1))
```

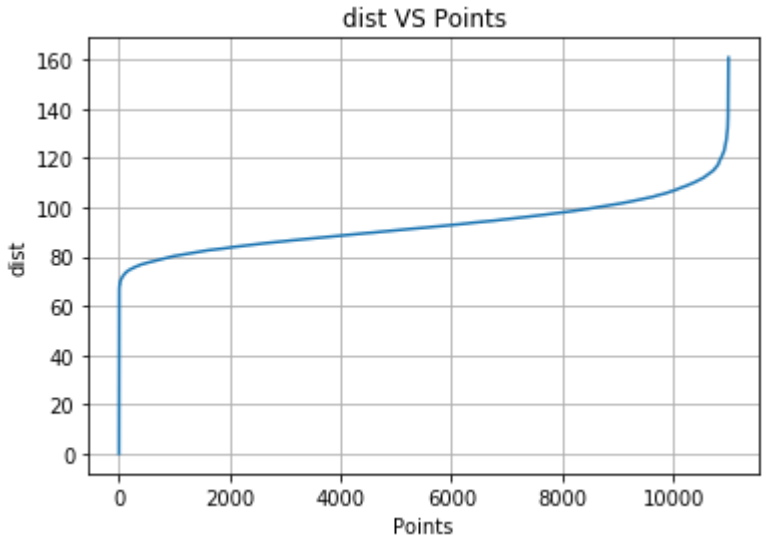
Out[49]:
```
array([[ 9.73445658],
       [ 9.75688429],
       [10.8408094 ],
       ...,
       [ 8.87598286],
       [ 9.55590757],
       [10.72392615]])
```

In [50]:
```
# X_tr=X.toarray()
# X_tr
```

In [51]:
```
min_points = 1500
from sklearn.preprocessing import StandardScaler
from sklearn.metrics.pairwise import euclidean_distances
X_trt=StandardScaler().fit_transform(X_trt)
distance=[]
for point in tqdm(X_trt):
    temp = euclidean_distances(X_trt, 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_trt))]
# 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%|██████████████████████████████████| 11000/11000 [1:13:29<00:00,  2.10it/s]
```



In [52]:
```
#we can see that point of inflexion is at eps=9
#In DBSCAN, the optimal value of 'min_samples' = (2*number of dimensions in the d
#The ideal value of 'min_samples' = 2*5000 = 10000

from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps=90,n_jobs=-1)
dbscan.fit(X_trt)
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}
```

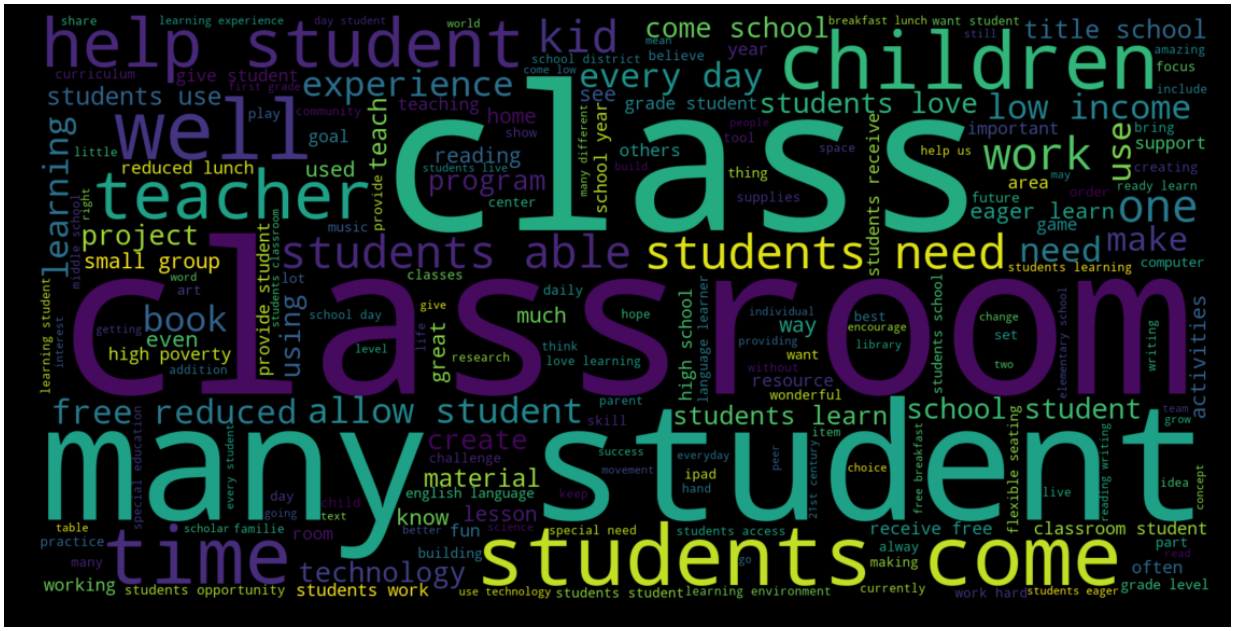
In [88]:
```
essays = X['essay'].values
```

In [89]:
```
#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 [90]:

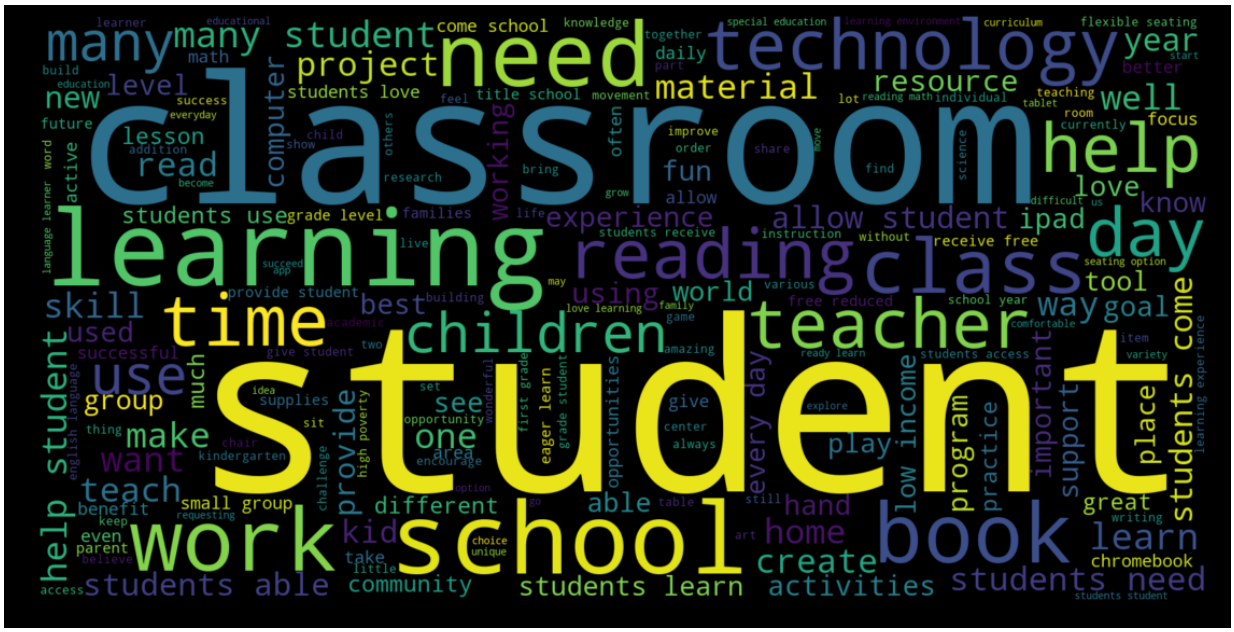
```
words=''
for i in cluster1:
    words+=str(i)
from wordcloud import WordCloud
wordcloud = WordCloud(width=1600, height=800).generate(words)


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



In [91]:
```
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:

Firstly we converted the reduced sparce matrix to dense using toarray().

Then we transformed the data to standard scalar.

Then we computed euclidean distance for every point to every other point and took the distance of min_pts.

Obtained the best eps to be 90 from the above graph b/w dist and points.

Then formed clusters on noise points and non-noise points.

Printed the essays in each of the two clusters.

Then printed the wordcloud.