1. Importing Packages

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
```

2. Loading Data

In [2]:

```
project_data = pd.read_csv('D:\\train_data.csv',nrows = 10000)
resource_data = pd.read_csv('D:\\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 (10000, 17)
==================================================
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 'school_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [4]:

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

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

In [5]:

```
resource_data.head()
```

Out[5]:

	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
2	p069063	Cory Stories: A Kid's Book About Living With Adhd	1	8.45
3	p069063	Dixon Ticonderoga Wood-Cased #2 HB Pencils, Bo...	2	13.59
4	p069063	EDUCATIONAL INSIGHTS FLUORESCENT LIGHT FILTERS...	3	24.95

In [6]:

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

Out[6]:

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

In [7]:

```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[7]:

	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 [8]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-groups-in-one-step
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index()
price_data.head(2)
```

Out[8]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [9]:

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

In [10]:

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

Out[10]:

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

In [11]:

```
project_data['teacher_prefix'] = project_data['teacher_prefix'].replace(np.NaN,'Mrs.')
```

3. Text Preprocessing

Concatenating all essay text

In [12]:

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

3.1. Preprocessing Essay text

In [13]:

```
# printing some random essays.
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[9999])
print("="*50)
```

```
My students are English learners that are working on English as their second or third languages. W
e are a melting pot of refugees, immigrants, and native-born Americans bringing the gift of langua
ge to our school. \r\n\r\n We have over 24 languages represented in our English Learner program wi
th students at every level of mastery.  We also have over 40 countries represented with the
families within 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 your language are the limits o
f your world.\"-Ludwig Wittgenstein  Our English learner's have a strong support system at home th
at begs for more resources.  Many times our parents are learning to read and speak English along s
ide of their children.  Sometimes this creates barriers for parents to be able to help their child
learn phonetics, letter recognition, and other reading skills.\r\n\r\nBy providing these dvd's and
players, students are able to continue their mastery of the English language even if no one at hom
```

players, students are able to continue their mastery of the English language even if no one at hom
e 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 En
glish Learner Teacher and will be sent home regularly to watch. The videos are to help the child
develop early reading skills.\r\n\r\nParents that do not have 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 ed
ucational 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 l
east most of the time. At our school, 97.3% of the students receive free or reduced price lunch. O
f the 560 students, 97.3% are minority students. \r\nThe school has a vibrant community that loves
to get together and celebrate. Around Halloween there is a whole school parade to show off the bea
utiful costumes that students wear. On Cinco de Mayo we put on a big festival with crafts made by
the students, dances, and games. At the end of the year the school hosts a carnival to celebrate t
he hard work put in during the school year, with a dunk tank being the most popular activity.My st
udents will use these five brightly colored Hokki stools in place of regular, stationary, 4-legged
chairs. As I will only have a total of ten in the classroom and not enough for each student to hav
e an individual one, they will be used in a variety of ways. During independent reading time they
will be used as special chairs students will each use on occasion. I will utilize them in place of
chairs at my small group tables during math and reading times. The rest of the day they will be us
ed by the students who need the highest amount of movement in their life in order to stay 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 stools we already have. When the students are sitting i
n 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 ta
ken. There are always students who head over to the kidney table to get one of the stools who are
disappointed as there are not enough of them. \r\n\r\nWe ask a lot of students to sit for 7 hours
a day. The Hokki stools will be a compromise that allow my students to do desk work and move at th
e same time. These stools will help students to meet their 60 minutes a day of movement by
allowing 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
==========
How do you remember your days of school? Was it in a sterile environment with plain walls, rows of
desks, and a teacher in front of the room? A typical day in our room is nothing like that. I work
hard to create a warm inviting themed room for my students look forward to coming to each day.\r\n
\r\nMy class is made up of 28 wonderfully unique boys and girls of mixed races in
Arkansas.\r\nThey attend a Title I school, which means there is a high enough percentage of free a
nd reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very uniq
ue as there are no walls separating the classrooms. These 9 and 10 year-old students are very eage
r learners; they are like sponges, absorbing all the information and experiences and keep on wanti
ng more.With these resources such as the comfy red throw pillows and the whimsical nautical hangin
g decor and the blue fish nets, I will be able to help create the mood in our classroom setting to
be one of a themed nautical environment. Creating a classroom environment is very important in the
success in each and every child's education. The nautical photo props will be used with each child
as they step foot into our classroom for the first time on Meet the Teacher evening. I'll take pic
tures of each child with them, have them developed, and then hung in our classroom ready for their
first day of 4th grade. This kind gesture will set the tone before even the first day of school!
The nautical thank you cards will be used throughout the year by the students as they create thank
you cards to their team groups.\r\n\r\nYour generous donations will help me to help make our
classroom a fun, inviting, learning environment from day one.\r\n\r\nIt costs lost of money out of
my own pocket on resources to get our classroom ready. Please consider helping with this project t
o make our new school year a very successful one. Thank you!nannan
==========
\"Creative Greatness\" is this school year's mantra to inspire my students to reach for the stars.
I'm excited about ushering in an enthusiasm and passion for growth in the visual arts department a
nd inspiring students to consider and apply the purpose of art outside of the classroom.
\r\n\r\nMy art students and art club members are not just \"taking\" art class, but are using thei
r creativity to engage in school-wide beautification projects and community initiatives. Help us t
o explore a greater variety of art media and technology in my Art 1 classes to ignite student's i
nterest in furthering their studies in art. Our large student body limits funding to the arts, so
charitable donations are crucial to our growth into Advanced Placement and College and Career Read
iness programs in the arts.Our class will create personalized and unique interactive notebooks to
encourage the development of independent learners and writers. Interactive notebooks are not just
used for class notes, but also for daily learning activities that require students to process the
information presented in class and then organize the content in a manner that will reinforce their
learning. \r\nInteractive Notebooks are a cross curricular tool that supports literacy in all cont
ent areas. In our art class, these notebooks are used not only as an affordable sketchbook option,
but also as an \"all things art\" guide that students can continue to reference throughout the sch
ool year and as they continue studies of more advanced art courses. We use our interactive noteboo
ks to write art critiques in response to viewing the works of famous artists and to write art stat
ements in response to the student's personal artwork. We also use interactive notebooks to build v
ocabulary skills with engaging activities to learn about the elements and principles of art to go
far beyond just defining the terms. Students are required to chose thinking maps that best organi
ze the information presented in the lesson to teach lifelong skills of literacy and note-taking. \
r\nStudents' interest in using interactive notebooks is positively impacted when they are able to
be creative and personalize the look of their notebooks. Engagement will no doubt be dramatically
increased with fun and colorful notebook covers and pages for each lesson. With this note-taking p

increased with fun and colorful notebook covers and pages for each lesson. With this note-taking p
rocess, students will learn organization, color coding, summarizing, and other important skills wh
ile creating personalized portfolios of their individual learning that they can reference
throughout the year.nannan
==

In [14]:

```
# 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 [15]:

```
sent = decontracted(project_data['essay'].values[9999])
print(sent)
print("="*50)
```

\"Creative Greatness\" is this school year is mantra to inspire my students to reach for the stars
. I am excited about ushering in an enthusiasm and passion for growth in the visual arts
department and inspiring students to consider and apply the purpose of art outside of the classroo
m. \r\n\r\nMy art students and art club members are not just \"taking\" art class, but are using t
heir creativity to engage in school-wide beautification projects and community initiatives. Help u
s to explore a greater variety of art media and technology in my Art 1 classes to ignite student
is interest in furthering their studies in art. Our large student body limits funding to the arts,
so charitable donations are crucial to our growth into Advanced Placement and College and Career R
eadiness programs in the arts.Our class will create personalized and unique interactive notebooks
to encourage the development of independent learners and writers. Interactive notebooks are not ju
st used for class notes, but also for daily learning activities that require students to process t
he information presented in class and then organize the content in a manner that will reinforce th
eir learning. \r\nInteractive Notebooks are a cross curricular tool that supports literacy in all
content areas. In our art class, these notebooks are used not only as an affordable sketchbook opt
ion, but also as an \"all things art\" guide that students can continue to reference throughout th
e school year and as they continue studies of more advanced art courses. We use our interactive no
tebooks to write art critiques in response to viewing the works of famous artists and to write art
statements in response to the student is personal artwork. We also use interactive notebooks to bu
ild vocabulary skills with engaging activities to learn about the elements and principles of art t
o go far beyond just defining the terms. Students are required to chose thinking maps that best o
rganize the information presented in the lesson to teach lifelong skills of literacy and note-taki
ng. \r\nStudents' interest in using interactive notebooks is positively impacted when they are abl
e to be creative and personalize the look of their notebooks. Engagement will no doubt be
dramatically increased with fun and colorful notebook covers and pages for each lesson. With this
note-taking process, students will learn organization, color coding, summarizing, and other
important skills while creating personalized portfolios of their individual learning that they can
reference throughout the year.nannan
==

In [16]:

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

 Creative Greatness is this school year is mantra to inspire my students to reach for the stars.
I am excited about ushering in an enthusiasm and passion for growth in the visual arts department
and inspiring students to consider and apply the purpose of art outside of the classroom.   My a

```
rt students and art club members are not just  taking  art class, but are using their creativity t
o engage in school-wide beautification projects and community initiatives. Help us to explore a gr
eater variety of art media and technology in my Art 1  classes to ignite student is interest in fu
rthering their studies in art. Our large student body limits funding to the arts, so charitable
donations are crucial to our growth into Advanced Placement and College and Career Readiness progr
ams in the arts.Our class will create personalized and unique interactive notebooks to encourage t
he development of independent learners and writers. Interactive notebooks are not just used for cl
ass notes, but also for daily learning activities that require students to process the information
presented in class and then organize the content in a manner that will reinforce their learning.
Interactive Notebooks are a cross curricular tool that supports literacy in all content areas. In
our art class, these notebooks are used not only as an affordable sketchbook option, but also as a
n  all things art  guide that students can continue to reference throughout the school year and as
they continue studies of more advanced art courses. We use our interactive notebooks to write art
critiques in response to viewing the works of famous artists and to write art statements in
response to the student is personal artwork. We also use interactive notebooks to build vocabulary
skills with engaging activities to learn about the elements and principles of art to go far beyond
just defining the terms.  Students are required to chose thinking maps that best organize the info
rmation presented in the lesson to teach lifelong skills of literacy and note-taking.   Students'
interest in using interactive notebooks is positively impacted when they are able to be creative a
nd personalize the look of their notebooks. Engagement will no doubt be dramatically increased
with fun and colorful notebook covers and pages for each lesson. With this note-taking process, st
udents will learn organization, color coding, summarizing, and other important skills while
creating personalized portfolios of their individual learning that they can reference throughout t
he year.nannan
```

In [17]:

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

```
 Creative Greatness is this school year is mantra to inspire my students to reach for the stars I
am excited about ushering in an enthusiasm and passion for growth in the visual arts department an
d inspiring students to consider and apply the purpose of art outside of the classroom My art stud
ents and art club members are not just taking art class but are using their creativity to engage i
n school wide beautification projects and community initiatives Help us to explore a greater varie
ty of art media and technology in my Art 1 classes to ignite student is interest in furthering the
ir studies in art Our large student body limits funding to the arts so charitable donations are cr
ucial to our growth into Advanced Placement and College and Career Readiness programs in the arts
Our class will create personalized and unique interactive notebooks to encourage the development o
f independent learners and writers Interactive notebooks are not just used for class notes but als
o for daily learning activities that require students to process the information presented in clas
s and then organize the content in a manner that will reinforce their learning Interactive Noteboo
ks are a cross curricular tool that supports literacy in all content areas In our art class these
notebooks are used not only as an affordable sketchbook option but also as an all things art guide
that students can continue to reference throughout the school year and as they continue studies of
more advanced art courses We use our interactive notebooks to write art critiques in response to v
iewing the works of famous artists and to write art statements in response to the student is
personal artwork We also use interactive notebooks to build vocabulary skills with engaging
activities to learn about the elements and principles of art to go far beyond just defining the te
rms Students are required to chose thinking maps that best organize the information presented in t
he lesson to teach lifelong skills of literacy and note taking Students interest in using interact
ive notebooks is positively impacted when they are able to be creative and personalize the look of
their notebooks Engagement will no doubt be dramatically increased with fun and colorful notebook
covers and pages for each lesson With this note taking process students will learn organization co
lor coding summarizing and other important skills while creating personalized portfolios of their
individual learning that they can reference throughout the year nannan
```

In [18]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
```

```
'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [19]:

```
# 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 = sent.lower()
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.strip())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 10000/10000 [00:
12<00:00, 810.47it/s]
```

In [20]:

```
# after preprocesing
preprocessed_essays[9999]
```

Out[20]:

```
'creative greatness school year mantra inspire students reach stars excited ushering enthusiasm pa
ssion growth visual arts department inspiring students consider apply purpose art outside
classroom art students art club members not taking art class using creativity engage school wide b
eautification projects community initiatives help us explore greater variety art media technology
art 1 classes ignite student interest furthering studies art large student body limits funding art
s charitable donations crucial growth advanced placement college career readiness programs arts cl
ass create personalized unique interactive notebooks encourage development independent learners wr
iters interactive notebooks not used class notes also daily learning activities require students p
rocess information presented class organize content manner reinforce learning interactive
notebooks cross curricular tool supports literacy content areas art class notebooks used not affor
dable sketchbook option also things art guide students continue reference throughout school year c
ontinue studies advanced art courses use interactive notebooks write art critiques response viewin
g works famous artists write art statements response student personal artwork also use interactive
notebooks build vocabulary skills engaging activities learn elements principles art go far beyond
defining terms students required chose thinking maps best organize information presented lesson te
ach lifelong skills literacy note taking students interest using interactive notebooks positively
impacted able creative personalize look notebooks engagement no doubt dramatically increased fun c
olorful notebook covers pages lesson note taking process students learn organization color coding
summarizing important skills creating personalized portfolios individual learning reference
throughout year nannan'
```

In [21]:

```
project_data['preprocessed_essays'] = preprocessed_essays
project_data.drop(['essay'], axis=1, inplace=True)
project_data.head(2)
```

Out[21]:

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

3.2. Preprocessing Title text

In [22]:

```
# printing some random essays.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[5000])
print("="*50)
print(project_data['project_title'].values[9999])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
Bouncing Our Wiggles and Worries Away!
==================================================
Note Your Ordinary Notebook!
==================================================
```

In [23]:

```
# 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 [24]:

```
title = decontracted(project_data['project_title'].values[9999])
print(title)
print("="*50)
```

```
Note Your Ordinary Notebook!
==================================================
```

In [25]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python/
title = title.replace('\\r', ' ')
title = title.replace('\\"', ' ')
title = title.replace('\\n', ' ')
print(title)
```

```
Note Your Ordinary Notebook!
```

In [26]:

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

```
Note Your Ordinary Notebook
```

In [27]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've",
\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"]
```

In [28]:

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

```
100%|██████████████████████████████████████████████████████████████████████████| 10000/10000
[00:00<00:00, 17103.76it/s]
```

In [29]:

```
# after preprocesing
preprocessed_titles[9999]
```

Out[29]:

```
'note ordinary notebook'
```

In [30]:

```
project_data['preprocessed_titles'] = preprocessed_titles
project_data.drop(['project_title'], axis=1, inplace=True)
project_data.head(2)
```

Out[30]:

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

In [31]:

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

Out[31]:

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

4. Preprocessing of Categorical data

4.1. Preprocessing project_grade_category

In [32]:

```
project_grade_clean_category = []

for i in range(len(project_data)):
    a = project_data["project_grade_category"][i].replace(" ", "_").replace("-", "_")
    project_grade_clean_category.append(a)
```

In [33]:

```
project_grade_clean_category[0:5]
```

Out[33]:

```
['Grades_PreK_2', 'Grades_6_8', 'Grades_6_8', 'Grades_PreK_2', 'Grades_PreK_2']
```

In [34]:

```
project_data['project_grade_clean_category'] = project_grade_clean_category
```

```
project_data.drop(['project_grade_category'], axis=1, inplace=True)
project_data.head(2)
```

Out[34]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_subject_ca
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	Literacy & L
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	History & Civics,

4.2. Preprocessing project_subject_categories

In [35]:

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

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

In [36]:

```
cat_list[0:5]
```

Out[36]:

```
['Literacy_Language',
 'History_Civics Health_Sports',
 'Health_Sports',
 'Literacy_Language Math_Science',
 'Math_Science']
```

In [37]:

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

Out[37]:

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

Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_subject_su

4.3. Preprocessing project_subject_subcategories

In [38]:

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

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

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

In [39]:

```
sub_cat_list[0:5]
```

Out[39]:

```
['ESL Literacy',
 'Civics_Government TeamSports',
 'Health_Wellness TeamSports',
 'Literacy Mathematics',
 'Mathematics']
```

In [40]:

```
project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)
project_data.head(2)
```

Out[40]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_essay_1
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	My students are English learners that are work...
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Our students arrive to our school eager to lea...

5. Sentiment score's of each of the essay

In [41]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

In [42]:

```
analyser = SentimentIntensityAnalyzer()
```

In [43]:

```
neg = []
pos = []
neu = []
compound = []

for a in tqdm(project_data["preprocessed_essays"]) :
    b = analyser.polarity_scores(a)['neg']
    c = analyser.polarity_scores(a)['pos']
    d = analyser.polarity_scores(a)['neu']
    e = analyser.polarity_scores(a)['compound']
    neg.append(b)
    pos.append(c)
    neu.append(d)
    compound.append(e)
```

```
100%|██████████████████████████████████████████████████████████████████████████████| 10000/10000 [02
:01<00:00, 82.45it/s]
```

In [44]:

```
project_data["pos"] = pos
```

In [45]:

```
project_data["neg"] = neg
```

In [46]:

```
project_data["neu"] = neu
```

In [47]:

```
project_data["compound"] = compound
```

In [48]:

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

Out[48]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_essay_1
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	My students are English learners that are work...
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Our students arrive to our school eager to lea...

2 rows × 24 columns

6. Counting number of words in the combine essays

In [49]:

```
essay_count = []
for word in project_data['preprocessed_essays']:
    a = len(word.split())
```

```
    b = str(a)
    essay_count.append(b)
```

In [50]:

```
essay_count[9999]
```

Out[50]:

'236'

In [51]:

```
project_data['number_of_words_in_essays'] = essay_count
project_data.head(2)
```

Out[51]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_essay_1
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	My students are English learners that are work...
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Our students arrive to our school eager to lea...

2 rows × 25 columns

7. Counting number of words in the title

In [52]:

```
title_count = []
for word in project_data['preprocessed_titles']:
    a = len(word.split())
    b = str(a)
    title_count.append(b)
```

In [53]:

```
title_count[9999]
```

Out[53]:

'3'

In [54]:

```
project_data['number_of_words_in_the_title'] = title_count
project_data.head(2)
```

Out[54]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_essay_1
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN	2016-12-05 13:43:57	My students are English learners that are work...
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL	2016-10-25 09:22:10	Our students arrive to our school eager to lea...

2 rows × 26 columns

8. Concatenating preprocessed_essays and preprocessed_titles in variable text

In [55]:

```
# https://stackoverflow.com/questions/39291499/how-to-concatenate-multiple-
# column-values-into-a-single-column-in-panda-datafram
text = project_data['preprocessed_essays'].map(str)+ ' ' + project_data['preprocessed_titles']
```

In [56]:

```
text.head()
```

Out[56]:

```
0    students english learners working english seco...
1    students arrive school eager learn polite gene...
2    true champions not always ones win guts mia ha...
3    work unique school filled esl english second l...
4    second grade classroom next year made around 2...
dtype: object
```

In [57]:

```
text.shape
```

Out[57]:

```
(10000,)
```

----------Sample example----------

In [58]:

```
text_sample = "abc def ijk pqr" , "pqr klm opq" , "lmn pqr xyz abc def pqr abc"
```

In [60]:

```
top_words_sample = "abc" , "pqr","def"
```

In [67]:

```
from tqdm import tqdm
window_size = 2
co_occurrence_matrix = np.zeros((3,3))
for row in tqdm(text_sample):
    words_in_row = row.split()
    for index,word in enumerate(words_in_row):
        if word in top_words_sample:
            for j in range(max(index-window_size,0),min(index+window_size,len(words_in_row)-1) + 1):
                if words_in_row[j] in top_words_sample:
                    co_occurrence_matrix[top_words_sample.index(word),top_words_sample.index(words_
in_row[j])] += 1
                else:
                    continue
        else:
            continue
```

```
100%|██████████| 3/3
[00:00<00:00, 3008.11it/s]
```

In [68]:

```
co_occurrence_matrix
```

Out[68]:

```
array([[3., 3., 3.],
       [3., 4., 2.],
       [3., 2., 2.]])
```

9. Applying TFIDF vectorizer on text

In [69]:

```
tfidf_vect = TfidfVectorizer()
text_tfidf = tfidf_vect.fit_transform(text)
```

In [70]:

```
text_tfidf.shape
```

Out[70]:

```
(10000, 23166)
```

In [71]:

```
sorted_features = np.argsort(tfidf_vect.idf_)
features = tfidf_vect.get_feature_names()

top_features = [features[i] for i in sorted_features[:2000]]
```

In [72]:

```
len(top_features)
```

Out[72]:

```
2000
```

In [73]:

```
top_features[0:10]
```

Out[73]:

```
['students',
 'nannan',
 'school',
 'learning',
 'classroom',
 'not',
 'learn',
 'help',
 'many',
 'need']
```

In [74]:

```
# Creating an empty Dataframe with column names only
df = pd.DataFrame(index =[x for x in top_features],columns=[x for x in top_features])
```

In [75]:

```
df.head()
```

Out[75]:

	students	nannan	school	learning	classroom	not	learn	help	many	need	...	site	knows	indoor	circle	discussing

	students	nannan	school	learning	classroom	not	learn	help	many	need	...	site	knows	indoor	circle	discussing
students	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
nannan	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
school	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
learning	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
classroom	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN

5 rows × 2000 columns

10. Computing the Co-occurrence matrix

In [77]:

```
from tqdm import tqdm
window_size = 5
co_occurrence_matrix = np.zeros((2000,2000))
for row in tqdm(text):
    words_in_row = row.split()
    for index,word in enumerate(words_in_row):
        if word in top_features:
            for j in range(max(index-window_size,0),min(index+window_size,len(words_in_row)-1) + 1):
                if words_in_row[j] in top_features:
                    co_occurrence_matrix[top_features.index(word),top_features.index(words_in_row[
])] += 1
                else:
                    continue
        else:
            continue
```

```
100%|██████████████████████████████████████████████████████████████████████████| 10000/10000 [06
:58<00:00, 26.53it/s]
```

In [78]:

```
co_occurrence_matrix
```

Out[78]:

```
array([[8.7274e+04, 1.9910e+03, 1.2924e+04, ..., 7.7000e+01, 5.1000e+01,
        2.5000e+01],
       [1.9910e+03, 9.5660e+03, 5.0700e+02, ..., 0.0000e+00, 2.0000e+00,
        6.0000e+00],
       [1.2924e+04, 5.0700e+02, 2.9581e+04, ..., 2.7000e+01, 1.0000e+00,
        8.0000e+00],
       ...,
       [7.7000e+01, 0.0000e+00, 2.7000e+01, ..., 7.5000e+01, 0.0000e+00,
        0.0000e+00],
       [5.1000e+01, 2.0000e+00, 1.0000e+00, ..., 0.0000e+00, 1.0500e+02,
        0.0000e+00],
       [2.5000e+01, 6.0000e+00, 8.0000e+00, ..., 0.0000e+00, 0.0000e+00,
        7.9000e+01]])
```

11. Applying truncated svd on Co-occurrence matrix

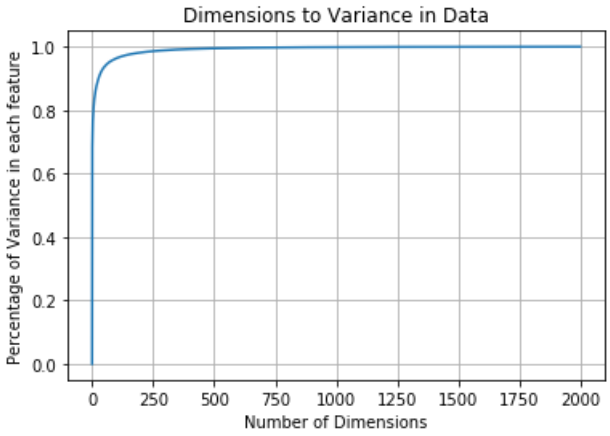
In [156]:

```
from sklearn.decomposition import TruncatedSVD
feature_number = [i for i in range(2000)]
Variance_sum = []
for i in tqdm(feature_number):
    svd = TruncatedSVD(n_components = i , random_state = 42)
    svd.fit(co_occurrence_matrix)
    Variance_sum.append(svd.explained_variance_ratio_.sum())
```

```
100%|██████████████████████████████████████████████████████████████████████████| 2000/2000
[4:36:23<00:00, 19.08s/it]
```

In [81]:

```
plt.figure(figsize=(6, 4))
plt.xlabel("Number of Dimensions")
plt.ylabel("Percentage of Variance in each feature")
plt.title("Dimensions to Variance in Data")
plt.plot(feature_number,Variance_sum)
plt.grid(True)
plt.show()
```



In [82]:

```
svd = TruncatedSVD(n_components = 250)
co_occurrence_svd = svd.fit_transform(co_occurrence_matrix)
```

In [83]:

```
co_occurrence_svd.shape
```

Out[83]:

```
(2000, 250)
```

In [84]:

```
len(co_occurrence_svd[0])
```

Out[84]:

```
250
```

In [85]:

```
len(co_occurrence_svd)
```

Out[85]:

```
2000
```

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

In [86]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(project_data,
project_data['project_is_approved'], test_size=0.33, stratify = project_data['project_is_approved'
])
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=y_train)
```

13. Dropping Target values from Train, Test and CV set

In [87]:

```
X_train.drop(['project_is_approved'], axis=1, inplace=True)
X_test.drop(['project_is_approved'], axis=1, inplace=True)
X_cv.drop(['project_is_approved'], axis=1, inplace=True)
```

In [88]:

```
print(X_train.shape)
print(X_test.shape)
print(X_cv.shape)
```

```
(4489, 25)
(3300, 25)
(2211, 25)
```

In [89]:

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

Out[89]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_essay_
4177	52551	p097305	2c1fb5bb511cdf3ca0b6a793c38661bb	Mr.	OR	2016-09-01 11:46:27	My fifth grad students a hungry f knowing
661	103747	p091163	dee51e804d70b6be40a5cb36b33d9265	Mr.	NY	2016-08-24 13:05:55	\"Mr. Pierre, I a ready to lea now!\", one

2 rows × 25 columns

In [90]:

```
y_train.head()
```

Out[90]:

```
4177    1
661     1
391     0
7472    1
1636    0
Name: project_is_approved, dtype: int64
```

14. Encoding Categorical Data

14.1. One Hot Encoding of clean_categories

In [91]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(X_train['clean_categories'].values)
print(vectorizer.get_feature_names())


categories_one_hot_Xtrain = vectorizer.transform(X_train['clean_categories'].values)
```

```
categories_one_hot_Xtest = vectorizer.transform(X_test['clean_categories'].values)
categories_one_hot_Xcv = vectorizer.transform(X_cv['clean_categories'].values)
print("Shape of matrix after one hot encodig ",categories_one_hot_Xtrain.shape)
print("Shape of matrix after one hot encodig ",categories_one_hot_Xtest.shape)
print("Shape of matrix after one hot encodig ",categories_one_hot_Xcv.shape)
```

```
['AppliedLearning', 'Care_Hunger', 'Health_Sports', 'History_Civics', 'Literacy_Language',
'Math_Science', 'Music_Arts', 'SpecialNeeds', 'Warmth']
Shape of matrix after one hot encodig  (4489, 9)
Shape of matrix after one hot encodig  (3300, 9)
Shape of matrix after one hot encodig  (2211, 9)
```

14.2. One Hot Encoding of clean_subcategories

In [92]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(X_train['clean_subcategories'].values)
print(vectorizer.get_feature_names())


sub_categories_one_hot_Xtrain = vectorizer.transform(X_train['clean_subcategories'].values)
sub_categories_one_hot_Xtest = vectorizer.transform(X_test['clean_subcategories'].values)
sub_categories_one_hot_Xcv = vectorizer.transform(X_cv['clean_subcategories'].values)

print("Shape of matrix after one hot encoding ",sub_categories_one_hot_Xtrain.shape)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_Xtest.shape)
print("Shape of matrix after one hot encodig ",sub_categories_one_hot_Xcv.shape)
```

```
['AppliedSciences', 'Care_Hunger', 'CharacterEducation', 'Civics_Government',
'College_CareerPrep', 'CommunityService', 'ESL', 'EarlyDevelopment', 'Economics',
'EnvironmentalScience', 'Extracurricular', 'FinancialLiteracy', 'ForeignLanguages', 'Gym_Fitness',
'Health_LifeScience', 'Health_Wellness', 'History_Geography', 'Literacy', 'Literature_Writing', 'M
athematics', 'Music', 'NutritionEducation', 'Other', 'ParentInvolvement', 'PerformingArts', 'Socia
lSciences', 'SpecialNeeds', 'TeamSports', 'VisualArts', 'Warmth']
Shape of matrix after one hot encoding  (4489, 30)
Shape of matrix after one hot encodig  (3300, 30)
Shape of matrix after one hot encodig  (2211, 30)
```

14.3. One Hot Encoding of school_state

In [93]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['school_state'].values)
print(vectorizer.get_feature_names())


school_state_one_hot_Xtrain = vectorizer.transform(X_train['school_state'].values)
school_state_one_hot_Xtest = vectorizer.transform(X_test['school_state'].values)
school_state_one_hot_Xcv = vectorizer.transform(X_cv['school_state'].values)

print("Shape of matrix after one hot encoding ",school_state_one_hot_Xtrain.shape)
print("Shape of matrix after one hot encoding ",school_state_one_hot_Xtest.shape)
print("Shape of matrix after one hot encoding ",school_state_one_hot_Xcv.shape)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'IA', 'ID', 'IL', 'IN', 'K
S', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO', 'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM',
'NV', 'NY', 'OH', 'OK', 'OR', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV
', 'WY']
Shape of matrix after one hot encoding  (4489, 51)
Shape of matrix after one hot encoding  (3300, 51)
Shape of matrix after one hot encoding  (2211, 51)
```

14.4. One Hot Encoding of teacher_prefix

14.4. One Hot Encoding of teacher_prefix

In [94]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['teacher_prefix'].values)
print(vectorizer.get_feature_names())


teacher_prefix_one_hot_Xtrain = vectorizer.transform(X_train['teacher_prefix'].values)
teacher_prefix_one_hot_Xtest = vectorizer.transform(X_test['teacher_prefix'].values)
teacher_prefix_one_hot_Xcv = vectorizer.transform(X_cv['teacher_prefix'].values)

print("Shape of matrix after one hot encoding ",teacher_prefix_one_hot_Xtrain.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_one_hot_Xtest.shape)
print("Shape of matrix after one hot encoding ",teacher_prefix_one_hot_Xcv.shape)
```

```
['Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encoding  (4489, 4)
Shape of matrix after one hot encoding  (3300, 4)
Shape of matrix after one hot encoding  (2211, 4)
```

14.5. One Hot Encoding of project_grade_clean_category

In [95]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer = CountVectorizer(lowercase=False, binary=True)
vectorizer.fit(project_data['project_grade_clean_category'].values)
print(vectorizer.get_feature_names())


grade_one_hot_Xtrain = vectorizer.transform(X_train['project_grade_clean_category'].values)
grade_one_hot_Xtest = vectorizer.transform(X_test['project_grade_clean_category'].values)
grade_one_hot_Xcv = vectorizer.transform(X_cv['project_grade_clean_category'].values)

print("Shape of matrix after one hot encoding ",grade_one_hot_Xtrain.shape)
print("Shape of matrix after one hot encoding ",grade_one_hot_Xtest.shape)
print("Shape of matrix after one hot encoding ",grade_one_hot_Xcv.shape)
```

```
['Grades_3_5', 'Grades_6_8', 'Grades_9_12', 'Grades_PreK_2']
Shape of matrix after one hot encoding  (4489, 4)
Shape of matrix after one hot encoding  (3300, 4)
Shape of matrix after one hot encoding  (2211, 4)
```

15. Encoding of Text Data

Avg-W2V on Essays and Titles(from SVD Matrix)

15.1. Average Word2Vec encoding of preprocessed_essays on Train Data

In [98]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_essays_Xtrain = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_train['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(250) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in top_features:
            i = top_features.index(word)
            vector += co_occurrence_svd[i]
            cnt_words += 1
```

```
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_essays_Xtrain.append(vector)

print(len(avg_w2v_vectors_essays_Xtrain))
print(len(avg_w2v_vectors_essays_Xtrain[2]))
```

```
100%|████████████████████████████████████████████████████████████████████████| 4489/4489
[00:14<00:00, 317.14it/s]
```

```
4489
250
```

In [99]:

```
average_w2v_on_essay_Xtrain = np.vstack(avg_w2v_vectors_essays_Xtrain)
print(average_w2v_on_essay_Xtrain.shape)
```

```
(4489, 250)
```

15.2. Average Word2Vec encoding of preprocessed_essays on Test Data

In [100]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_essays_Xtest = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_test['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(250) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in top_features:
            i = top_features.index(word)
            vector += co_occurrence_svd[i]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_essays_Xtest.append(vector)

print(len(avg_w2v_vectors_essays_Xtest))
print(len(avg_w2v_vectors_essays_Xtest[2]))
```

```
100%|████████████████████████████████████████████████████████████████████████| 3300/3300
[00:11<00:00, 276.38it/s]
```

```
3300
250
```

In [101]:

```
average_w2v_on_essay_Xtest = np.vstack(avg_w2v_vectors_essays_Xtest)
print(average_w2v_on_essay_Xtest.shape)
```

```
(3300, 250)
```

15.3. Average Word2Vec encoding of preprocessed_essays on CV Data

In [102]:

```
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_essays_Xcv = []; # the avg-w2v for each sentence/review is stored in this list
for sentence in tqdm(X_cv['preprocessed_essays'].values): # for each review/sentence
```

```
for sentence in tqdm(X_cv['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(250) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in sentence.split(): # for each word in a review/sentence
        if word in top_features:
            i = top_features.index(word)
            vector += co_occurrence_svd[i]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_essays_Xcv.append(vector)

print(len(avg_w2v_vectors_essays_Xcv))
print(len(avg_w2v_vectors_essays_Xcv[2]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 2211/2211
[00:06<00:00, 336.74it/s]
```

```
2211
250
```

In [103]:

```
average_w2v_on_essay_Xcv = np.vstack(avg_w2v_vectors_essays_Xcv)
print(average_w2v_on_essay_Xcv.shape)
```

```
(2211, 250)
```

15.4. Average Word2Vec encoding of preprocessed_titles on Train Data

In [104]:

```
#t-title
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_titles_Xtrain = []; # the avg-w2v for each sentence/review is stored in this list
for t in tqdm(X_train['preprocessed_titles'].values): # for each review/sentence
    vector = np.zeros(250) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in t.split(): # for each word in a review/sentence
        if word in top_features:
            i = top_features.index(word)
            vector += co_occurrence_svd[i]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_Xtrain.append(vector)

print(len(avg_w2v_vectors_titles_Xtrain))
print(len(avg_w2v_vectors_titles_Xtrain[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 4489/4489
[00:00<00:00, 7620.39it/s]
```

```
4489
250
```

In [105]:

```
average_w2v_on_titles_Xtrain = np.vstack(avg_w2v_vectors_titles_Xtrain)
print(average_w2v_on_titles_Xtrain.shape)
```

```
(4489, 250)
```

15.5. Average Word2Vec encoding of preprocessed_titles on Test

15.5. Average Word2Vec encoding of preprocessed_titles on Test Data

In [106]:

```
#t-title
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_titles_Xtest = []; # the avg-w2v for each sentence/review is stored in this list
for t in tqdm(X_test['preprocessed_titles'].values): # for each review/sentence
    vector = np.zeros(250) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in t.split(): # for each word in a review/sentence
        if word in top_features:
            i = top_features.index(word)
            vector += co_occurrence_svd[i]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_Xtest.append(vector)

print(len(avg_w2v_vectors_titles_Xtest))
print(len(avg_w2v_vectors_titles_Xtest[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 3300/3300
[00:00<00:00, 6965.73it/s]
```

```
3300
250
```

In [107]:

```
average_w2v_on_titles_Xtest = np.vstack(avg_w2v_vectors_titles_Xtest)
print(average_w2v_on_titles_Xtest.shape)
```

```
(3300, 250)
```

15.6. Average Word2Vec encoding of preprocessed_titles on CV Data

In [108]:

```
#t-title
# average Word2Vec
# compute average word2vec for each review.
avg_w2v_vectors_titles_Xcv = []; # the avg-w2v for each sentence/review is stored in this list
for t in tqdm(X_cv['preprocessed_titles'].values): # for each review/sentence
    vector = np.zeros(250) # as word vectors are of zero length
    cnt_words =0; # num of words with a valid vector in the sentence/review
    for word in t.split(): # for each word in a review/sentence
        if word in top_features:
            i = top_features.index(word)
            vector += co_occurrence_svd[i]
            cnt_words += 1
    if cnt_words != 0:
        vector /= cnt_words
    avg_w2v_vectors_titles_Xcv.append(vector)

print(len(avg_w2v_vectors_titles_Xcv))
print(len(avg_w2v_vectors_titles_Xcv[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 2211/2211
[00:00<00:00, 6231.75it/s]
```

```
2211
250
```

In [109]:

```
average_w2v_on_titles_Xcv = np.vstack(avg_w2v_vectors_titles_Xcv)
print(average_w2v_on_titles_Xcv.shape)
```

```
(2211, 250)
```

16. Encoding of Numerical Data

16.1. Encoding of price on Train,Test and CV data

In [110]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
from sklearn.preprocessing import MinMaxScaler


scalar = MinMaxScaler()

price_standardized_Xtrain = scalar.fit_transform(X_train['price'].values.reshape(-1, 1))
price_standardized_Xtest = scalar.transform(X_test['price'].values.reshape(-1,1))
price_standardized_Xcv = scalar.transform(X_cv['price'].values.reshape(-1, 1))
```

In [111]:

```
price_standardized_Xtrain
```

Out[111]:

```
array([[0.02647456],
       [0.11074319],
       [0.01724988],
       ...,
       [0.07946771],
       [0.05772203],
       [0.00509903]])
```

In [112]:

```
print(price_standardized_Xtrain.shape)
print(price_standardized_Xtest.shape)
print(price_standardized_Xcv.shape)
```

```
(4489, 1)
(3300, 1)
(2211, 1)
```

16.2. Encoding of quantity on Train,Test and CV data

In [113]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
from sklearn.preprocessing import MinMaxScaler

scalar = MinMaxScaler()

quantity_standardized_Xtrain = scalar.fit_transform(X_train['quantity'].values.reshape(-1, 1))
quantity_standardized_Xtest = scalar.transform(X_test['quantity'].values.reshape(-1, 1))
quantity_standardized_Xcv = scalar.transform(X_cv['quantity'].values.reshape(-1, 1))
```

In [114]:

```
quantity_standardized_Xtrain
```

Out[114]:

```
array([[0.        ],
       [0.00172414],
       [0.00172414],
       ...,
       [0.06724138],
       [0.00172414],
       [0.08103448]])
```

In [115]:

```
print(quantity_standardized_Xtrain.shape)
print(quantity_standardized_Xtest.shape)
print(quantity_standardized_Xcv.shape)
```

```
(4489, 1)
(3300, 1)
(2211, 1)
```

16.3. Encoding of teacher_number_of_previously_posted_projects on Train,Test and CV data

In [116]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
from sklearn.preprocessing import MinMaxScaler

scalar = MinMaxScaler()

# Now standardize the data with above maen and variance.
teacher_number_of_previously_posted_projects_standardized_Xtrain = scalar.fit_transform(X_train['t
eacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
teacher_number_of_previously_posted_projects_standardized_Xtest =
scalar.transform(X_test['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
teacher_number_of_previously_posted_projects_standardized_Xcv =
scalar.transform(X_cv['teacher_number_of_previously_posted_projects'].values.reshape(-1, 1))
```

In [117]:

```
teacher_number_of_previously_posted_projects_standardized_Xtrain
```

Out[117]:

```
array([[0.00529101],
       [0.02645503],
       [0.0026455 ],
       ...,
       [0.15608466],
       [0.        ],
       [0.00529101]])
```

In [118]:

```
print(teacher_number_of_previously_posted_projects_standardized_Xtrain.shape)
print(teacher_number_of_previously_posted_projects_standardized_Xtest.shape)
print(teacher_number_of_previously_posted_projects_standardized_Xcv.shape)
```

```
(4489, 1)
(3300, 1)
(2211, 1)
```

16.4. Encoding of pos on Train,Test and CV data

In [119]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
from sklearn.preprocessing import MinMaxScaler

scalar = MinMaxScaler()

essay_pos_standardized_Xtrain = scalar.fit_transform(X_train['pos'].values.reshape(-1, 1))
essay_pos_standardized_Xtest = scalar.transform(X_test['pos'].values.reshape(-1, 1))
essay_pos_standardized_Xcv = scalar.transform(X_cv['pos'].values.reshape(-1, 1))
```

In [120]:

```
essay_pos_standardized_Xtrain
```

Out[120]:

```
array([[0.63414634],
       [0.52157598],
       [0.33395872],
       ...,
       [0.58536585],
       [0.48968105],
       [0.37148218]])
```

In [121]:

```
print(essay_pos_standardized_Xtrain.shape)
print(essay_pos_standardized_Xtest.shape)
print(essay_pos_standardized_Xcv.shape)
```

```
(4489, 1)
(3300, 1)
(2211, 1)
```

16.5. Encoding of neg on Train,Test and CV data

In [122]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
from sklearn.preprocessing import MinMaxScaler

scalar = MinMaxScaler()

essay_neg_standardized_Xtrain = scalar.fit_transform(X_train['neg'].values.reshape(-1, 1))
essay_neg_standardized_Xtest = scalar.transform(X_test['neg'].values.reshape(-1, 1))
essay_neg_standardized_Xcv = scalar.transform(X_cv['neg'].values.reshape(-1, 1))
```

In [123]:

```
essay_neg_standardized_Xtrain
```

Out[123]:

```
array([[0.13061224],
       [0.37959184],
       [0.31428571],
       ...,
       [0.44489796],
       [0.30204082],
       [0.25714286]])
```

In [124]:

```
print(essay_neg_standardized_Xtrain.shape)
print(essay_neg_standardized_Xtest.shape)
print(essay_neg_standardized_Xcv.shape)
```

```
(4489, 1)
(3300, 1)
(2211, 1)
```

16.6. Encoding of neu on Train,Test and CV data

In [125]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
from sklearn.preprocessing import MinMaxScaler

scalar = MinMaxScaler()

essay_neu_standardized_Xtrain = scalar.fit_transform(X_train['neu'].values.reshape(-1, 1))
essay_neu_standardized_Xtest = scalar.transform(X_test['neu'].values.reshape(-1, 1))
essay_neu_standardized_Xcv = scalar.transform(X_cv['neu'].values.reshape(-1, 1))
```

In [126]:

```
essay_neu_standardized_Xtrain
```

Out[126]:

```
array([[0.3853211 ],
       [0.38165138],
       [0.59449541],
       ...,
       [0.28990826],
       [0.44587156],
       [0.58348624]])
```

In [127]:

```
print(essay_neu_standardized_Xtrain.shape)
print(essay_neu_standardized_Xtest.shape)
print(essay_neu_standardized_Xcv.shape)
```

```
(4489, 1)
(3300, 1)
(2211, 1)
```

16.7. Encoding of compound on Train,Test and CV data

In [128]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
from sklearn.preprocessing import MinMaxScaler

scalar = MinMaxScaler()

essay_compound_standardized_Xtrain = scalar.fit_transform(X_train['compound'].values.reshape(-1, 1)
)
essay_compound_standardized_Xtest = scalar.transform(X_test['compound'].values.reshape(-1, 1))
essay_compound_standardized_Xcv = scalar.transform(X_cv['compound'].values.reshape(-1, 1))
```

In [129]:

```
essay_compound_standardized_Xtrain
```

Out[129]:

```
array([[0.99754526],
       [0.99202209],
       [0.97719137],
       ...,
```

```
       ...,
       [0.99688043],
       [0.98302138],
       [0.9951928 ]])
```

In [130]:

```
print(essay_compound_standardized_Xtrain.shape)
print(essay_compound_standardized_Xtest.shape)
print(essay_compound_standardized_Xcv.shape)
```

```
(4489, 1)
(3300, 1)
(2211, 1)
```

16.8. Encoding of number_of_words_in_essays on Train,Test and CV data

In [131]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
from sklearn.preprocessing import MinMaxScaler

scalar = MinMaxScaler()

number_of_words_in_essays_standardized_Xtrain =
scalar.fit_transform(X_train['number_of_words_in_essays'].values.reshape(-1, 1))
number_of_words_in_essays_standardized_Xtest = scalar.transform(X_test['number_of_words_in_essays'
].values.reshape(-1, 1))
number_of_words_in_essays_standardized_Xcv = scalar.transform(X_cv['number_of_words_in_essays'].va
lues.reshape(-1, 1))
```

In [132]:

```
number_of_words_in_essays_standardized_Xtrain
```

Out[132]:

```
array([[0.28837209],
       [0.29767442],
       [0.3255814 ],
       ...,
       [0.46976744],
       [0.0372093 ],
       [0.73023256]])
```

In [133]:

```
print(number_of_words_in_essays_standardized_Xtrain.shape)
print(number_of_words_in_essays_standardized_Xtest.shape)
print(number_of_words_in_essays_standardized_Xcv.shape)
```

```
(4489, 1)
(3300, 1)
(2211, 1)
```

16.9. Encoding of number_of_words_in_the_title on Train,Test and CV data

In [134]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
# https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
from sklearn.preprocessing import MinMaxScaler
```

```
scalar = MinMaxScaler()

number_of_words_in_the_title_standardized_Xtrain =
scalar.fit_transform(X_train['number_of_words_in_the_title'].values.reshape(-1, 1))
number_of_words_in_the_title_standardized_Xtest =
scalar.transform(X_test['number_of_words_in_the_title'].values.reshape(-1, 1))
number_of_words_in_the_title_standardized_Xcv =
scalar.transform(X_cv['number_of_words_in_the_title'].values.reshape(-1, 1))
```

In [135]:

```
number_of_words_in_the_title_standardized_Xtrain
```

Out[135]:

```
array([[0.5],
       [0.4],
       [0.6],
       ...,
       [0.4],
       [0.3],
       [0.2]])
```

In [136]:

```
print(number_of_words_in_the_title_standardized_Xtrain.shape)
print(number_of_words_in_the_title_standardized_Xtest.shape)
print(number_of_words_in_the_title_standardized_Xcv.shape)
```

```
(4489, 1)
(3300, 1)
(2211, 1)
```

17. Creating Sets of Data for Training Model

In [137]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
Xtrain1 =
hstack((categories_one_hot_Xtrain,sub_categories_one_hot_Xtrain,school_state_one_hot_Xtrain,teache
r_prefix_one_hot_Xtrain,grade_one_hot_Xtrain,price_standardized_Xtrain,quantity_standardized_Xtrain
,teacher_number_of_previously_posted_projects_standardized_Xtrain,essay_pos_standardized_Xtrain,es
say_neg_standardized_Xtrain,essay_neu_standardized_Xtrain,essay_compound_standardized_Xtrain,numbe
r_of_words_in_essays_standardized_Xtrain,number_of_words_in_the_title_standardized_Xtrain,average_
w2v_on_essay_Xtrain,average_w2v_on_titles_Xtrain)).tocsr()
Xtest1 = hstack(( categories_one_hot_Xtest,sub_categories_one_hot_Xtest,school_state_one_hot_Xtest
,teacher_prefix_one_hot_Xtest,grade_one_hot_Xtest,price_standardized_Xtest,quantity_standardized_Xt
est,teacher_number_of_previously_posted_projects_standardized_Xtest,essay_pos_standardized_Xtest,e
ssay_neg_standardized_Xtest,essay_neu_standardized_Xtest,essay_compound_standardized_Xtest,number_
of_words_in_essays_standardized_Xtest,number_of_words_in_the_title_standardized_Xtest,average_w2v_o
n_essay_Xtest,average_w2v_on_titles_Xtest)).tocsr()
Xcv1 =
hstack((categories_one_hot_Xcv,sub_categories_one_hot_Xcv,school_state_one_hot_Xcv,teacher_prefix_o
ne_hot_Xcv,grade_one_hot_Xcv,price_standardized_Xcv,quantity_standardized_Xcv,teacher_number_of_pre
viously_posted_projects_standardized_Xcv,essay_pos_standardized_Xcv,essay_neg_standardized_Xcv,ess
ay_neu_standardized_Xcv,essay_compound_standardized_Xcv,number_of_words_in_essays_standardized_Xcv
,number_of_words_in_the_title_standardized_Xcv,average_w2v_on_essay_Xcv,average_w2v_on_titles_Xcv)
).tocsr()

print(Xtrain1.shape,y_train.shape)
print(Xtest1.shape,y_test.shape)
print(Xcv1.shape,y_cv.shape)
```

```
(4489, 607) (4489,)
(3300, 607) (3300,)
(2211, 607) (2211,)
```

18. Applying XGBOOST

18. Applying XGBOOST

In [138]:

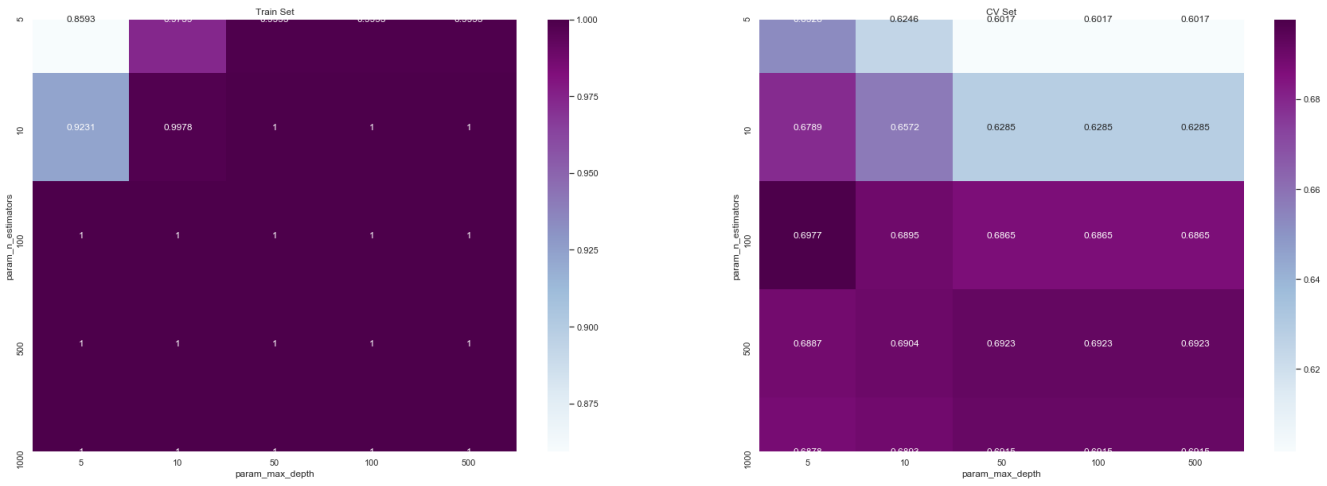
```
# https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV
import xgboost
import seaborn as sb


XG = xgboost.XGBClassifier(scale_pos_weight=1, n_jobs = -1)
parameters = {'max_depth':(5,10,50,100,500) , 'n_estimators':(5,10,100,500,1000)}
XGB = GridSearchCV(XG, parameters, cv=3, scoring='roc_auc',return_train_score=True)
XGB.fit(Xtrain1, y_train)
print('Best estimator', XGB.best_estimator_)
print('Best score', XGB.best_score_)
```

```
Best estimator XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=5,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=-1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
Best score 0.6976841581107617
```

18.1. Finding The Best Hyperparameter "max_depth" and "n_estimators"

In [139]:

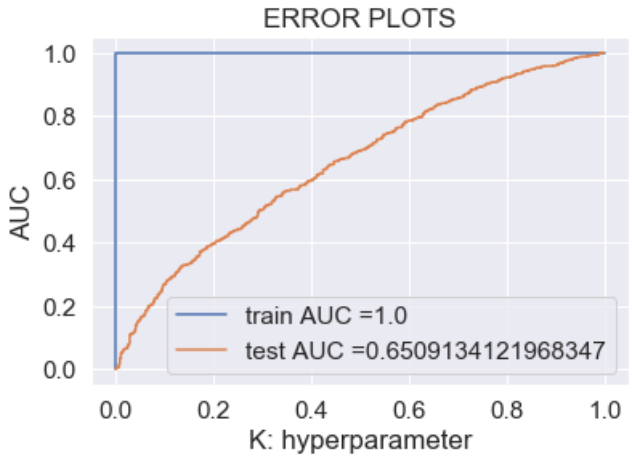
```
XGB = pd.DataFrame.from_dict(XGB.cv_results_)
max_scores_avg = XGB.groupby(['param_n_estimators',
                              'param_max_depth']).max()
max_scores_avg = max_scores_avg.unstack()[['mean_test_score', 'mean_train_score']]
#https://towardsdatascience.com/using-3d-visualizations-to-tune-hyperparameters-of-ml-models-with-python-ba2885eab2e9
import seaborn as sns; sns.set()

fig, ax = plt.subplots(1,2, figsize=(30,10))
sns.heatmap(max_scores_avg.mean_train_score, annot = True, fmt='.4g',cmap= "BuPu", ax=ax[0])
sns.heatmap(max_scores_avg.mean_test_score, annot = True, fmt='.4g',cmap="BuPu", ax=ax[1])
ax[0].set_title('Train Set')
ax[1].set_title('CV Set')
plt.show()
```



18.2. Testing the performance of the model on test data, plotting ROC Curves

In [140]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


XG1 = xgboost.XGBClassifier(scale_pos_weight=1,n_jobs = -1, max_depth = 5 ,n_estimators =  100)
XG1.fit(Xtrain1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred_avg = XG1.predict_proba(Xtrain1)[:,1]
y_test_pred_avg = XG1.predict_proba(Xtest1)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_avg)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_avg)
sns.set(font_scale = 1.4)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



18.3. Building Confusion matrix

In [141]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [142]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
```

```
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_avg, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_avg, best_t)))
```

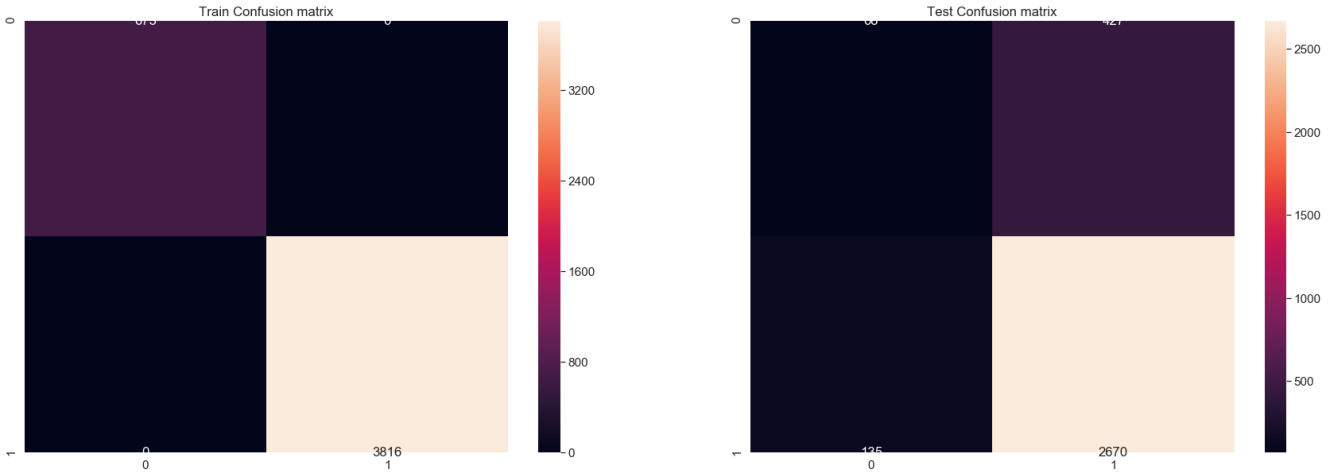
```
========================================================================================================

the maximum value of tpr*(1-fpr) 1.0 for threshold 0.707
Train confusion matrix
[[ 673    0]
 [   0 3816]]
Test confusion matrix
[[  68  427]
 [ 135 2670]]
```

In [143]:

```
confusion_matrix_train_avg = pd.DataFrame(confusion_matrix(y_train,
predict_with_best_t(y_train_pred_avg, best_t)))
confusion_matrix_test_avg = pd.DataFrame(confusion_matrix(y_test,
predict_with_best_t(y_test_pred_avg, best_t)))
```

In [144]:

```
import seaborn as sns
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(30,10))
# sns.set(font_scale = 4)
sns.heatmap(confusion_matrix_train_avg,annot = True ,ax = axes[0],fmt='g')
sns.heatmap(confusion_matrix_test_avg,annot = True , ax = axes[1],fmt = 'g')
axes[0].set_title('Train Confusion matrix')
axes[1].set_title('Test Confusion matrix')
plt.show()
```

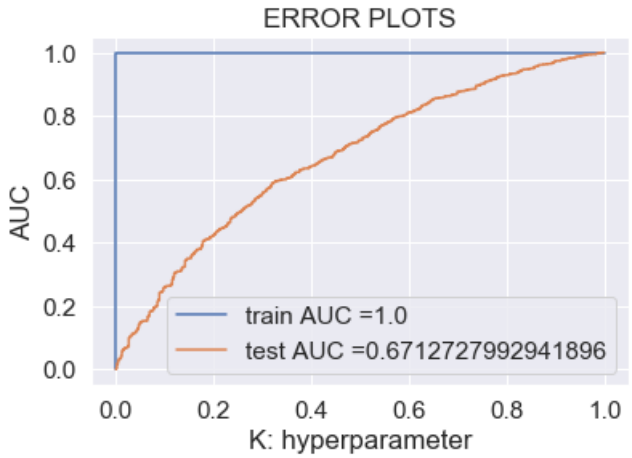


Randomly trying different max_depth and n_estimators values for getting best Test AUC

In [151]:

```
# https://scikit-
learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html#sklearn.metrics.roc_curve
from sklearn.metrics import roc_curve, auc


XG1 = xgboost.XGBClassifier(scale_pos_weight=1,n_jobs = -1, max_depth = 20 ,n_estimators =  100)
XG1.fit(Xtrain1, y_train)
# roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the positive
class
# not the predicted outputs

y_train_pred_avg = XG1.predict_proba(Xtrain1)[:,1]
y_test_pred_avg = XG1.predict_proba(Xtest1)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_avg)
```

```
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_avg)
sns.set(font_scale = 1.4)
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



19. Confusion matrix

In [152]:

```
print("="*100)
from sklearn.metrics import confusion_matrix
best_t = find_best_threshold(tr_thresholds, train_fpr, train_tpr)
print("Train confusion matrix")
print(confusion_matrix(y_train, predict_with_best_t(y_train_pred_avg, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_avg, best_t)))
```

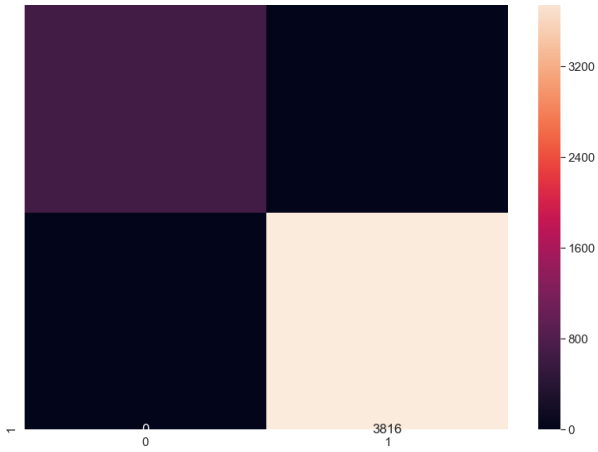
```
====================================================================================================

the maximum value of tpr*(1-fpr) 1.0 for threshold 0.969
Train confusion matrix
[[ 673    0]
 [   0 3816]]
Test confusion matrix
[[ 277  218]
 [ 928 1877]]
```

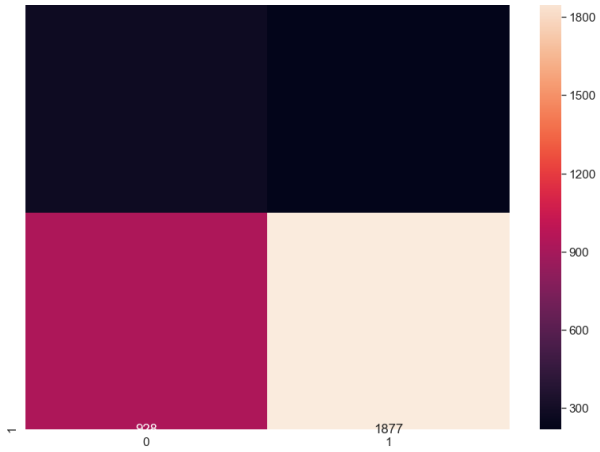
In [153]:

```
confusion_matrix_train_avg = pd.DataFrame(confusion_matrix(y_train,
predict_with_best_t(y_train_pred_avg, best_t)))
confusion_matrix_test_avg = pd.DataFrame(confusion_matrix(y_test,
predict_with_best_t(y_test_pred_avg, best_t)))
```

In [154]:

```
import seaborn as sns
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(30,10))
# sns.set(font_scale = 4)
sns.heatmap(confusion_matrix_train_avg,annot = True ,ax = axes[0],fmt='g')
sns.heatmap(confusion_matrix_test_avg,annot = True , ax = axes[1],fmt = 'g')
axes[0].set_title('Train Confusion matrix')
axes[1].set_title('Test Confusion matrix')
plt.show()
```





By trying out with different values of max_depth and n_estimators we found out that max_depth = 20,n_estimators = 100 increased our Test AUC score which is 0.6712 and also increased our True Negatives and decresed the False Positives.But still not a good model as it is still overfitting but better than the previous one.

20. Conclusion

In [155]:

```
# http://zetcode.com/python/prettytable/
from prettytable import PrettyTable

#If you get a ModuleNotFoundError error , install prettytable using: pip3 install prettytable

x = PrettyTable()

x.field_names = ["Vectorizer", "Model", "Hyper parameter", "Train AUC","Test AUC"]

x.add_row(["Average w2v", "XGBOOST","(max_depth = 5,n_estimators = 100)",1,0.6509])
x.add_row(["Average w2v", "XGBOOST","(max_depth = 20,n_estimators = 100)",1,0.6712])


print(x)
```

```
+-------------+---------+-------------------------------------+-----------+----------+
|  Vectorizer |  Model  |           Hyper parameter           | Train AUC | Test AUC |
+-------------+---------+-------------------------------------+-----------+----------+
| Average w2v | XGBOOST |  (max_depth = 5,n_estimators = 100) |     1     |  0.6509  |
| Average w2v | XGBOOST | (max_depth = 20,n_estimators = 100) |     1     |  0.6712  |
+-------------+---------+-------------------------------------+-----------+----------+
```

In [ ]:

```

```