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

In [4]:

```
print("Number of data points in 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()
```

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
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

In [7]:

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

In [8]:

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

3. Text Preprocessing

3.1. Concatenating all essay text

In [9]:

```
# 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)
```

3.2. Preprocessing Essay text

In [10]:

```
# 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[20000])
print("="*50)
print(project_data['essay'].values[49999])
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
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
```

\r\nMy class is made up of 26 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 and reduced-price lunch to qualify. Our school is an \"open classroom\" concept, which is very unique as there are no walls separating the classrooms. These 9 and 10 year-old students are very eager learners; they are like sponges, absorbing all the information and experiences and keep on wanting more.With these resources such as the comfy red throw pillows and the whimsical nautical hanging 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 pictures 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 to make our new school year a very successful one. Thank you!nannan
==
My kindergarten students have varied disabilities ranging from speech and language delays, cognitive delays, gross/fine motor delays, to autism. They are eager beavers and always strive to work their hardest working past their limitations. \r\n\r\nThe materials we have are the ones I seek out for my students. I teach in a Title I school where most of the students receive free or reduced price lunch. Despite their disabilities and limitations, my students love coming to school and come eager to learn and explore.Have you ever felt like you had ants in your pants and you needed to groove and move as you were in a meeting? This is how my kids feel all the time. The want to be able to move as they learn or so they say.Wobble chairs are the answer and I love then because they develop their core, which enhances gross motor and in Turn fine motor skills. \r\nThey also want to learn through games, my kids don't want to sit and do worksheets. They want to learn to count by jumping and playing. Physical engagement is the key to our success. The number toss and color and shape mats can make that happen. My students will forget they are doing work and just have the fun a 6 year old deserves.nannan
==
We have GRIT! If you want to meet tenacious, respectful seven year olds with growth mindsets, you need to come to our classroom. We give hugs, high-fives, and compliments! We Begin with the End in Mind and work hard everyday to reach our goals.\r\n\r\nWe don't believe in making excuses, but there are times in life when you just need to ask for help. As a classroom teacher in a low-income/high poverty school district, my 2nd grade students face real-life struggles both in and out of the classroom. Even though, as a visitor to my classroom, you wouldn't know the daily struggle for some of them. I ask you. How can you learn with your belly growling? How can I provide the absolute best learning environment when we do not have the money to buy research-based materials? \r\n\r\n\"Education is not the filling of a pail, but the lighting of a fire,\" William Butler Yeats. We are not asking you to fill our pail with \"things,\"but to help provide resources to light the fire in young minds.Receiving books written by the same author will teach students how to develop their own Writer's Craft. It will inspire them to think about different ways established authors have developed successful text that appeal to various audiences. \r\n\r\nWe never forget our first love. My mother read the Berenstain Bears series to me when I was five and I fell in love with the Berenstain family. She took me to the public library every week and I would hunt for books written by Stan and Jan Berenstain. Next, was the curious monkey and the man in the yellow hat, Curious George! Thank you Margaret and H.A. Rey for creating a series that captured my heart and attention. \r\n\r\nAs a teacher, it is my hope and dream to inspire the students in my classroom to find their first love in reading. Help me help them to discover writer's craft, go on adventures in their minds, and develop a tenacious love for reading for the sake of reading.nannan
==

In [11]:

```
# 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 [12]:

```
sent = decontracted(project_data['essay'].values[16499])
print(sent)
print("="*50)
```

```
I teach an amazing, energetic, engaged, and kind group of 5th grade students in an inner city high
poverty public school in Indianapolis. Many of my students have parents who work odd hours and hav
e limited time to spend with their wonderfully talented children. My students work hard in class g
iving 110% with everything that they do. They persevere through difficult topics, enjoy being
engaged in their hands-on activities, and they love to laugh while learning. I set high
expectation for my students. They understand that true, authentic learning takes hard work,
dedication, and requires them to take ownership over their education. My goal for my students is t
o leave my class as life long learners. The students work hard to overcome all obstacles in their
path to meet and grow past my expectations.My students love being active while they are learning a
nd wiggling while they are working. I am lucky enough to have one Hokki stools in my classroom. Sa
dly, one is not enough to reach all my students. My students love to use the Hokki stools while th
ey learn and want more! One of my students suggested that I write a project since I \"only have on
e, and we need more.\" These stools help my amazing kiddos get focused while engaging their core t
o keep them happy and healthy.\r\n\r\nMy students love to wiggle so they can not only focus on the
ir work, but engage in a healthy lifestyle.\r\n\r\n\r\nThe Hokki stools would allow my students to
continue to be active throughout the day whether they are in small groups or working at their own
seat.\r\nnannan
==================================================
```

In [13]:

```
# \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)
```

```
I teach an amazing, energetic, engaged, and kind group of 5th grade students in an inner city high
poverty public school in Indianapolis. Many of my students have parents who work odd hours and hav
e limited time to spend with their wonderfully talented children. My students work hard in class g
iving 110% with everything that they do. They persevere through difficult topics, enjoy being
engaged in their hands-on activities, and they love to laugh while learning. I set high
expectation for my students. They understand that true, authentic learning takes hard work,
dedication, and requires them to take ownership over their education. My goal for my students is t
o leave my class as life long learners. The students work hard to overcome all obstacles in their
path to meet and grow past my expectations.My students love being active while they are learning a
nd wiggling while they are working. I am lucky enough to have one Hokki stools in my classroom. Sa
dly, one is not enough to reach all my students. My students love to use the Hokki stools while th
ey learn and want more! One of my students suggested that I write a project since I  only have one
, and we need more.  These stools help my amazing kiddos get focused while engaging their core to
keep them happy and healthy.    My students love to wiggle so they can not only focus on their wor
k, but engage in a healthy lifestyle.      The Hokki stools would allow my students to continue to
be active throughout the day whether they are in small groups or working at their own seat.
nannan
```

In [14]:

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

```
I teach an amazing energetic engaged and kind group of 5th grade students in an inner city high
poverty public school in Indianapolis Many of my students have parents who work odd hours and have
limited time to spend with their wonderfully talented children My students work hard in class givi
ng 110 with everything that they do They persevere through difficult topics enjoy being engaged in
their hands on activities and they love to laugh while learning I set high expectation for my stud
ents They understand that true authentic learning takes hard work dedication and requires them to
take ownership over their education My goal for my students is to leave my class as life long
learners The students work hard to overcome all obstacles in their path to meet and grow past my e
xpectations My students love being active while they are learning and wiggling while they are work
ing I am lucky enough to have one Hokki stools in my classroom Sadly one is not enough to reach al
l my students My students love to use the Hokki stools while they learn and want more One of my st
udents suggested that I write a project since I only have one and we need more These stools help m
y amazing kiddos get focused while engaging their core to keep them happy and healthy My students
love to wiggle so they can not only focus on their work but engage in a healthy lifestyle The Hokk
i stools would allow my students to continue to be active throughout the day whether they are in s
mall groups or working at their own seat nannan
```

In [15]:

```
# 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 [16]:

```
# 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%|████████████████████████████████████████████████████████████| 109248/109248
[01:32<00:00, 1179.85it/s]
```

In [17]:

```
# after preprocesing
preprocessed_essays[20000]
```

Out[17]:

```
'kindergarten students varied disabilities ranging speech language delays cognitive delays gross f
ine motor delays autism eager beavers always strive work hardest working past limitations
materials ones seek students teach title school students receive free reduced price lunch despite
disabilities limitations students love coming school come eager learn explore ever felt like ants
pants needed groove move meeting kids feel time want able move learn say wobble chairs answer love
develop core enhances gross motor turn fine motor skills also want learn games kids not want sit w
orksheets want learn count jumping playing physical engagement key success number toss color shape
mats make happen students forget work fun 6 year old deserves nannan'
```

In [18]:

```
project_data['preprocessed_essays'] = preprocessed_essays
```

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

Out[18]:

	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

3.4. Preprocessing Title text

In [19]:

```
# 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[20000])
print("="*50)
print(project_data['project_title'].values[49999])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Young Authors Through Reading
==================================================
```

In [20]:

```
# 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 [21]:

```
title = decontracted(project_data['project_title'].values[20000])
print(title)
print("="*50)
```

```
We Need To Move It While We Input It!
==================================================
```

In [22]:

```
# \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)
```

```
We Need To Move It While We Input It!
```

In [23]:

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

```
We Need To Move It While We Input It
```

In [24]:

```
# 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 [25]:

```
# 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%|██████████████████████████████████████████████████████████████████████████████| 109248/109248
[00:04<00:00, 27171.52it/s]
```

In [26]:

```
# after preprocesing
preprocessed_titles[20000]
```

Out[26]:

```
'need move input'
```

In [27]:

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

Out[27]:

	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 [28]:

```
project_data.head()
```

Out[28]:

	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
2	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	Grade
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	Grades P
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

4. Preprocessing of Categorical Data

4.1. Preprocessing project_grade_category

In [29]:

```
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 [30]:

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

Out[30]:

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

In [31]:

```
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[31]:

	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 [32]:

```
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 [33]:

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

Out[33]:

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

In [34]:

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

Out[34]:

	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

4.3. Preprocessing project_subject_subcategories

In [35]:

```
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 [36]:

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

Out[36]:

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

In [37]:

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

Out[37]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_essay_1

	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...

In [38]:

```
project_data.head()
```

Out[38]:

	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	21895	p182444	3465aaf82da834c0582ebd0ef8040ca0	Ms.	AZ	2016-08-31 12:03:56	\r\n\"True champions aren't always the ones th...
3	45	p246581	f3cb9bffbba169bef1a77b243e620b60	Mrs.	KY	2016-10-06 21:16:17	I work at a unique school filled with both ESL...
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Our second grade classroom next year will be m...

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

In [39]:

```
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)
```

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

In [40]:

```
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 [41]:

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

```
(49041, 19)
(36052, 19)
(24155, 19)
```

In [42]:

```
print(y_train.value_counts())
print(y_test.value_counts())
print(y_cv.value_counts())
```

```
1    41615
0     7426
Name: project_is_approved, dtype: int64
1    30593
0     5459
Name: project_is_approved, dtype: int64
1    20498
0     3657
Name: project_is_approved, dtype: int64
```

In [43]:

```
y_train.head()
```

Out[43]:

```
76697    1
86985    1
69537    1
53526    1
13327    0
Name: project_is_approved, dtype: int64
```

In [44]:

```
X_train.head()
```

Out[44]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_essay
76697	21155	p234745	0706f308596b2742a026ad372d7c8d60	Mr.	OK	2016-09-07 11:23:03	My students deserving, no a mediocre
86985	92431	p130795	9e71753459c670d51675b9a8ba5310b2	Mr.	WI	2017-01-26 23:53:30	We have talen young peo who si dance
69537	85545	p181247	4daaef228e9d6547e0f73028b15f4abc	Mrs.	MO	2016-11-11 15:17:11	I teach in a l income a where a lo m
53526	69222	p008234	87f99120cb983740fc9551f3b849658b	Mrs.	NV	2017-03-08 20:54:47	I am part of founding s that just ope
13327	81900	p190845	76d15b96b676e781a1022ae4ec2c2a3d	Ms.	FL	2016-12-13 09:22:13	My students incredi creative a expre

7. Encoding Categorical Data

7.1. One Hot Encoding of clean_categories

In [45]:

```
# # count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# from collections import Counter
# my_counter = Counter()
# for word in project_data['clean_categories'].values:
#     my_counter.update(word.split())
```

```
#     my_counter.update(word.split())

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

In [46]:

```
# we use count vectorizer to convert the values into one hot encoded features
from sklearn.feature_extraction.text import CountVectorizer
vectorizer1= CountVectorizer(lowercase=False, binary=True)
vectorizer1.fit(project_data['clean_categories'].values)
print(vectorizer1.get_feature_names())


categories_one_hot_Xtrain = vectorizer1.transform(X_train['clean_categories'].values)
categories_one_hot_Xtest = vectorizer1.transform(X_test['clean_categories'].values)
categories_one_hot_Xcv = vectorizer1.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  (49041, 9)
Shape of matrix after one hot encodig  (36052, 9)
Shape of matrix after one hot encodig  (24155, 9)
```

7.2. One Hot Encoding of clean_subcategories

In [47]:

```
# # count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# from collections import Counter
# my_counter = Counter()
# for word in project_data['clean_subcategories'].values:
#     my_counter.update(word.split())

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

In [48]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer2 = CountVectorizer(lowercase=False, binary=True)
vectorizer2.fit(project_data['clean_subcategories'].values)
print(vectorizer2.get_feature_names())


sub_categories_one_hot_Xtrain = vectorizer2.transform(X_train['clean_subcategories'].values)
sub_categories_one_hot_Xtest = vectorizer2.transform(X_test['clean_subcategories'].values)
sub_categories_one_hot_Xcv = vectorizer2.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  (49041, 30)
Shape of matrix after one hot encodig  (36052, 30)
Shape of matrix after one hot encodig  (24155, 30)
```

7.3. One Hot Encoding of school_state

In [49]:

```
# # count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# from collections import Counter
# my_counter = Counter()
# for word in project_data['school_state'].values:
#     my_counter.update(word.split())

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

In [50]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer3 = CountVectorizer(lowercase=False, binary=True)
vectorizer3.fit(project_data['school_state'].values)
print(vectorizer3.get_feature_names())


school_state_one_hot_Xtrain = vectorizer3.transform(X_train['school_state'].values)
school_state_one_hot_Xtest = vectorizer3.transform(X_test['school_state'].values)
school_state_one_hot_Xcv = vectorizer3.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  (49041, 51)
Shape of matrix after one hot encoding  (36052, 51)
Shape of matrix after one hot encoding  (24155, 51)
```

7.4. One Hot Encoding of teacher_prefix

In [51]:

```
# # count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# from collections import Counter
# my_counter = Counter()
# for word in project_data['teacher_prefix'].values:
#     my_counter.update(word.split())

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

In [52]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer4 = CountVectorizer(lowercase=False, binary=True)
vectorizer4.fit(project_data['teacher_prefix'].values)
print(vectorizer4.get_feature_names())


teacher_prefix_one_hot_Xtrain = vectorizer4.transform(X_train['teacher_prefix'].values)
teacher_prefix_one_hot_Xtest = vectorizer4.transform(X_test['teacher_prefix'].values)
teacher_prefix_one_hot_Xcv = vectorizer4.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)
```

```
['Dr', 'Mr', 'Mrs', 'Ms', 'Teacher']
Shape of matrix after one hot encoding  (49041, 5)
Shape of matrix after one hot encoding  (36052, 5)
```

Shape of matrix after one hot encoding (24155, 5)

7.5. One Hot Encoding of project_grade_clean_category

In [53]:

```
# # count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
# from collections import Counter
# my_counter = Counter()
# for word in project_data['project_grade_clean_category'].values:
#     my_counter.update(word.split())

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

In [54]:

```
# we use count vectorizer to convert the values into one hot encoded features
vectorizer5 = CountVectorizer(lowercase=False, binary=True)
vectorizer5.fit(project_data['project_grade_clean_category'].values)
print(vectorizer5.get_feature_names())


grade_one_hot_Xtrain = vectorizer5.transform(X_train['project_grade_clean_category'].values)
grade_one_hot_Xtest = vectorizer5.transform(X_test['project_grade_clean_category'].values)
grade_one_hot_Xcv = vectorizer5.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  (49041, 4)
Shape of matrix after one hot encoding  (36052, 4)
Shape of matrix after one hot encoding  (24155, 4)
```

8. Encoding of Text Data

8.1.1. BOW encoding of preprocessed_essays

In [55]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer6 = CountVectorizer(min_df=10)
text_bow_Xtrain = vectorizer6.fit_transform(X_train['preprocessed_essays'].values)
print("Shape of matrix after one hot encodig ",text_bow_Xtrain.shape)
text_bow_Xtest = vectorizer6.transform(X_test['preprocessed_essays'].values)
print("Shape of matrix after one hot encodig ",text_bow_Xtest.shape)
text_bow_Xcv = vectorizer6.transform(X_cv['preprocessed_essays'].values)
print("Shape of matrix after one hot encodig ",text_bow_Xcv.shape)
```

```
Shape of matrix after one hot encodig  (49041, 11997)
Shape of matrix after one hot encodig  (36052, 11997)
Shape of matrix after one hot encodig  (24155, 11997)
```

8.1.2. BOW encoding of preprocessed_titles

In [56]:

```
# We are considering only the words which appeared in at least 10 documents(rows or projects).
vectorizer7 = CountVectorizer(min_df=10)
title_bow_Xtrain = vectorizer7.fit_transform(X_train['preprocessed_titles'].values)
print("Shape of matrix after one hot encodig ",title_bow_Xtrain.shape)
title_bow_Xtest = vectorizer7.transform(X_test['preprocessed_titles'].values)
```

```
print("Shape of matrix after one hot encodig ",title_bow_Xtest.shape)
title_bow_Xcv = vectorizer7.transform(X_cv['preprocessed_titles'].values)
print("Shape of matrix after one hot encodig ",title_bow_Xcv.shape)
```

```
Shape of matrix after one hot encodig  (49041, 2019)
Shape of matrix after one hot encodig  (36052, 2019)
Shape of matrix after one hot encodig  (24155, 2019)
```

8.2.1. TFIDF encoding of preprocessed_essays

In [57]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer8 = TfidfVectorizer(min_df=10)
text_tfidf_Xtrain = vectorizer8.fit_transform(X_train['preprocessed_essays'].values)
print("Shape of matrix after one hot encodig ",text_tfidf_Xtrain.shape)
text_tfidf_Xtest = vectorizer8.transform(X_test['preprocessed_essays'].values)
print("Shape of matrix after one hot encodig ",text_tfidf_Xtest.shape)
text_tfidf_Xcv = vectorizer8.transform(X_cv['preprocessed_essays'].values)
print("Shape of matrix after one hot encodig ",text_tfidf_Xcv.shape)
```

```
Shape of matrix after one hot encodig  (49041, 11997)
Shape of matrix after one hot encodig  (36052, 11997)
Shape of matrix after one hot encodig  (24155, 11997)
```

8.2.2. TFIDF encoding of preprocessed_titles

In [58]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer9 = TfidfVectorizer(min_df=10)
title_tfidf_Xtrain = vectorizer9.fit_transform(X_train['preprocessed_titles'].values)
print("Shape of matrix after one hot encodig ",title_tfidf_Xtrain.shape)
title_tfidf_Xtest = vectorizer9.transform(X_test['preprocessed_titles'].values)
print("Shape of matrix after one hot encodig ",title_tfidf_Xtest.shape)
title_tfidf_Xcv = vectorizer9.transform(X_cv['preprocessed_titles'].values)
print("Shape of matrix after one hot encodig ",title_tfidf_Xcv.shape)
```

```
Shape of matrix after one hot encodig  (49041, 2019)
Shape of matrix after one hot encodig  (36052, 2019)
Shape of matrix after one hot encodig  (24155, 2019)
```

9. Encoding of Numerical Data

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

In [59]:

```
# 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 [60]:

```
price_standardized_Xtrain
```

Out[60]:

```
array([[0.04591262],
       [0.07213097],
       [0.00053009],
       ...,
       [0.01852808],
       [0.01342623],
       [0.03446772]])
```

In [61]:

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

```
(49041, 1)
(36052, 1)
(24155, 1)
```

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

In [62]:

```
# 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 [63]:

```
quantity_standardized_Xtrain
```

Out[63]:

```
array([[0.00889878],
       [0.00222469],
       [0.04338154],
       ...,
       [0.01446051],
       [0.        ],
       [0.00778643]])
```

In [64]:

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

```
(49041, 1)
(36052, 1)
(24155, 1)
```

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

In [65]:

```
# 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 [66]:

```
teacher_number_of_previously_posted_projects_standardized_Xtrain
```

Out[66]:

```
array([[0.00665188],
       [0.        ],
       [0.00886918],
       ...,
       [0.00443459],
       [0.00665188],
       [0.        ]])
```

In [67]:

```
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)
```

```
(49041, 1)
(36052, 1)
(24155, 1)
```

10. Printing Dimensions of all Preprocessed Data

In [68]:

```
print(categories_one_hot_Xtrain.shape)
print(categories_one_hot_Xtest.shape)
print(categories_one_hot_Xcv.shape)
print(sub_categories_one_hot_Xtrain.shape)
print(sub_categories_one_hot_Xtest.shape)
print(sub_categories_one_hot_Xcv.shape)
print(school_state_one_hot_Xtrain.shape)
print(school_state_one_hot_Xtest.shape)
print(school_state_one_hot_Xcv.shape)
print(teacher_prefix_one_hot_Xtrain.shape)
print(teacher_prefix_one_hot_Xtest.shape)
print(teacher_prefix_one_hot_Xcv.shape)
print(grade_one_hot_Xtrain.shape)
print(grade_one_hot_Xtest.shape)
print(grade_one_hot_Xcv.shape)
print(text_bow_Xtrain.shape)
print(text_bow_Xtest.shape)
print(text_bow_Xcv.shape)
print(title_bow_Xtrain.shape)
print(title_bow_Xtest.shape)
print(title_bow_Xcv.shape)
print(text_tfidf_Xtrain.shape)
print(text_tfidf_Xtest.shape)
print(text_tfidf_Xcv.shape)
print(title_tfidf_Xtrain.shape)
print(title_tfidf_Xtest.shape)
print(title_tfidf_Xcv.shape)
print(price_standardized_Xtrain.shape)
print(price_standardized_Xtest.shape)
print(price_standardized_Xcv.shape)
print(quantity_standardized_Xtrain.shape)
```

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

```
(49041, 9)
(36052, 9)
(24155, 9)
(49041, 30)
(36052, 30)
(24155, 30)
(49041, 51)
(36052, 51)
(24155, 51)
(49041, 5)
(36052, 5)
(24155, 5)
(49041, 4)
(36052, 4)
(24155, 4)
(49041, 11997)
(36052, 11997)
(24155, 11997)
(49041, 2019)
(36052, 2019)
(24155, 2019)
(49041, 11997)
(36052, 11997)
(24155, 11997)
(49041, 2019)
(36052, 2019)
(24155, 2019)
(49041, 1)
(36052, 1)
(24155, 1)
(49041, 1)
(36052, 1)
(24155, 1)
(49041, 1)
(36052, 1)
(24155, 1)
```

11. Creating Different Sets of Data for Training Model

Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW)

In [69]:

```
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,text_bow_Xtrain,title_bow_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_Xte
st,teacher_number_of_previously_posted_projects_standardized_Xtest,text_bow_Xtest,title_bow_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,text_bow_Xcv,title_bow_Xcv)).tocsr()

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

```
(49041, 14118) (49041,)
```

```
(36052, 14118) (36052,)
(24155, 14118) (24155,)
```

Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (TFIDF)

In [70]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
Xtrain2 =
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,text_tfidf_Xtrain,title_tfidf_Xtr
ain)).tocsr()
Xtest2 = 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_Xte
st,teacher_number_of_previously_posted_projects_standardized_Xtest,text_tfidf_Xtest,title_tfidf_Xte
st)).tocsr()
Xcv2 =
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,text_tfidf_Xcv,title_tfidf_Xcv)).tocsr()

print(Xtrain2.shape,y_train.shape)
print(Xtest2.shape,y_test.shape)
print(Xcv2.shape,y_cv.shape)
```

```
(49041, 14118) (49041,)
(36052, 14118) (36052,)
(24155, 14118) (24155,)
```

12. Appling Multinomial Naive Bayes on different kind of featurization

12.1. Applying Naive Bayes on BOW, SET 1

Function for predicting Target values Batchwise

In [72]:

```
# def batch_predict(clf, data):
#     # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the po
sitive class
#     # not the predicted outputs

#     y_data_pred = []
#     tr_loop = data.shape[0] - data.shape[0]%1000
#     # consider you X_tr shape is 49041, then your tr_loop will be 49041 - 49041%1000 = 49000
#     # in this for loop we will iterate unti the last 1000 multiplier
#     for i in range(0, tr_loop, 1000):
#         y_data_pred.extend(clf.predict_proba(data[i:i+1000])[:,1])
#     # we will be predicting for the last data points
#     if data.shape[0]%1000 !=0:
#         y_data_pred.extend(clf.predict_proba(data[tr_loop:])[:,1])

#     return y_data_pred
```

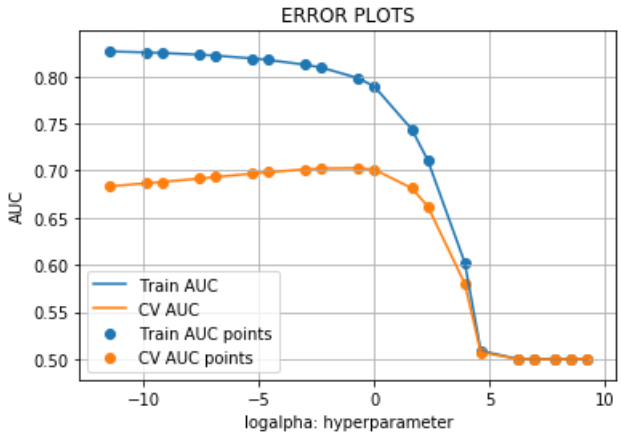
12.1.1. Finding The Best Hyperparameter "alpha"

In [73]:

```
import math
```

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

"""

train_auc = []
cv_auc = []
log_alpha = []
alpha = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100,
         500, 1000, 2500, 5000, 10000]
for i in tqdm(alpha):
    naive_bayes = MultinomialNB(alpha = i, class_prior = [0.5,0.5], fit_prior=True)
    naive_bayes.fit(Xtrain1, y_train)

    y_train_pred_bow = naive_bayes.predict_proba(Xtrain1)[:,1]
    y_cv_pred_bow = naive_bayes.predict_proba(Xcv1)[:,1]

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred_bow))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred_bow))

for a in tqdm(alpha):
    b = math.log(a)
    log_alpha.append(b)

plt.plot(log_alpha, train_auc, label='Train AUC')
plt.plot(log_alpha, cv_auc, label='CV AUC')

plt.scatter(log_alpha, train_auc, label='Train AUC points')
plt.scatter(log_alpha, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("logalpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```

```
100%|██████████████████████████████████████████████████████████████████████████████████████| 20/20
[00:04<00:00,  6.61it/s]
100%|██████████████████████████████████████████████████████████████████████████████████████| 20/20
[00:00<00:00, 19996.68it/s]
```



In [74]:

```
score_cv = [x for x in cv_auc]
optimal_alpha_cv = alpha[score_cv.index(max(score_cv))]
print("Maximum AUC score of cv is:" + ' ' + str(max(score_cv)))
print("Corresponding alpha value of cv is:",optimal_alpha_cv, '\n')
best_alpha_bow = optimal_alpha_cv
print(best_alpha_bow)
```

```
Maximum AUC score of cv is: 0.7027102466068239
Corresponding alpha value of cv is: 0.5

0.5
```

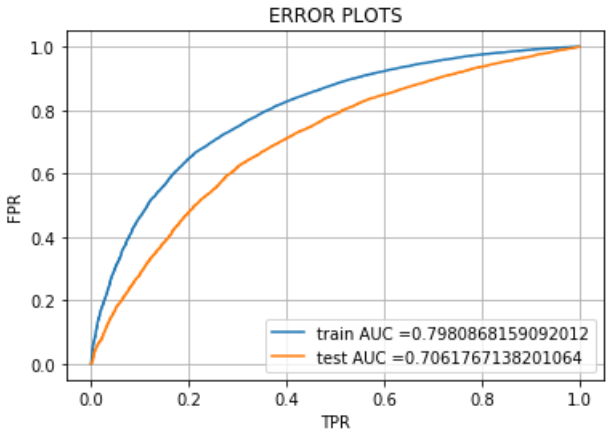
12.1.2. Testing the performance of the model on test data, plotting ROC Curves

In [75]:

```
# best_alpha_bow = 2
```

In [76]:

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


naive_bayes = MultinomialNB(alpha= best_alpha_bow, class_prior = [0.5,0.5], fit_prior=True)
naive_bayes.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_bow = naive_bayes.predict_proba(Xtrain1)[:,1]
y_test_pred_bow = naive_bayes.predict_proba(Xtest1)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_bow)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_bow)

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("TPR")
plt.ylabel("FPR")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



12.1.3. Building Confusion Matrix

In [77]:

```
# we are writing our own function for predict, with defined thresould
```

```
# 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 [78]:

```
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_bow, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_bow, best_t)))
```

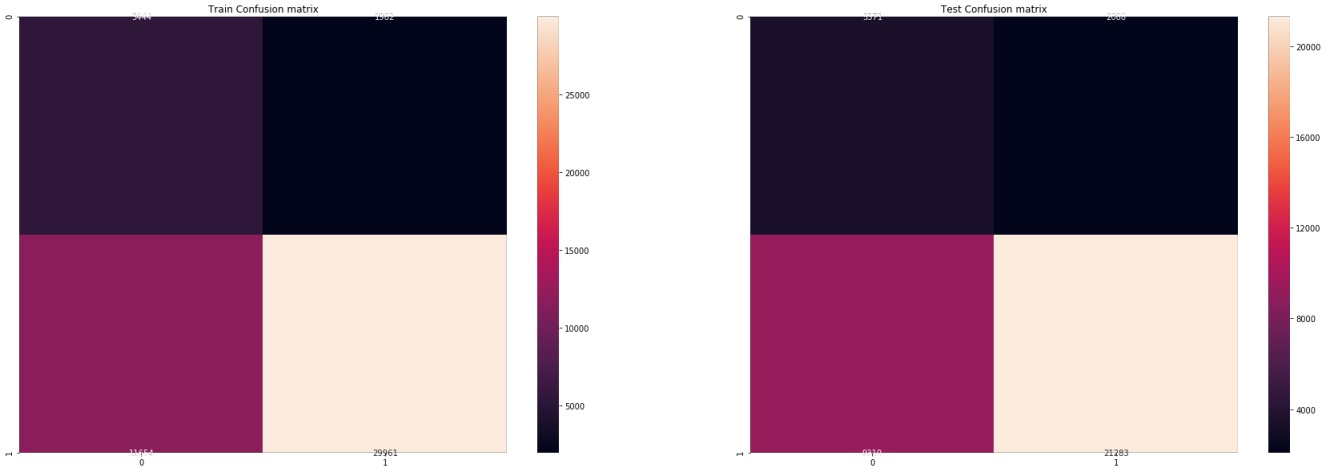
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.5278002325900545 for threshold 0.519
Train confusion matrix
[[ 5444  1982]
 [11654 29961]]
Test confusion matrix
[[ 3371  2088]
 [ 9310 21283]]
```

In [79]:

```
confusion_matrix_train_bow = pd.DataFrame(confusion_matrix(y_train,
predict_with_best_t(y_train_pred_bow, best_t)))
confusion_matrix_test_bow = pd.DataFrame(confusion_matrix(y_test,
predict_with_best_t(y_test_pred_bow, best_t)))
```

In [80]:

```
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_bow,annot = True ,ax = axes[0],fmt='g')
sns.heatmap(confusion_matrix_test_bow,annot = True , ax = axes[1],fmt = 'g')
axes[0].set_title('Train Confusion matrix')
axes[1].set_title('Test Confusion matrix')
plt.show()
```



12.1.4. Appending feature names to a list

In [72]:

```
set1_features_names = []
for a in vectorizer1.get_feature_names() :# clean categories
    set1_features_names.append(a)
for a in vectorizer2.get_feature_names() :# sub categoreis
    set1_features_names.append(a)
for a in vectorizer3.get_feature_names() :# school state
    set1_features_names.append(a)
for a in vectorizer4.get_feature_names() :# teacher prefix
    set1_features_names.append(a)
for a in vectorizer5.get_feature_names() :# grade categories
    set1_features_names.append(a)

set1_features_names.append('price')
set1_features_names.append('quantity')
set1_features_names.append('teacher_number_of_previously_posted_projects')

for a in vectorizer6.get_feature_names(): # essays bow
    set1_features_names.append(a)
for a in vectorizer7.get_feature_names(): # titles bow
    set1_features_names.append(a)
print( len(set1_features_names))
```

```
14118
```

12.1.5. Top 10 important features of negative class from SET 1

In [73]:

```
#BOW
import math
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score

naive_bayes = MultinomialNB(alpha = 0.5, class_prior = [0.5,0.5], fit_prior=True)# takes the k from
the i th list value
naive_bayes.fit(Xtrain1, y_train)# fit the model
```

Out[73]:

```
MultinomialNB(alpha=0.5, class_prior=[0.5, 0.5], fit_prior=True)
```

In [74]:

```
# now make a dictionary of all the probabilities fo the weights
set1_features_negative_probs = []
for a in range( len(set1_features_names)):
    set1_features_negative_probs.append(naive_bayes.feature_log_prob_[0,a])
print(len(set1_features_negative_probs))
```

```
14118
```

In [75]:

```
#top 10 negative features
final_bow_features = pd.DataFrame({'feature_probability_estimates_negative' :
set1_features_negative_probs, 'set1_feature_names_negative_label': set1_features_names})
a =final_bow_features.sort_values(by = ['feature_probability_estimates_negative'], ascending =
False)

a.head(10)
```

Out[75]:

	feature_probability_estimates_negative	set1_feature_names_negative_label
10454	-3.013000	students
9496	-4.101784	school
6301	-4.433578	learning
2124	-4.566558	classroom
7322	-4.777556	not
6297	-4.791060	learn
5213	-4.818509	help
7155	-4.986400	nannan
6667	-5.015207	many
7202	-5.130512	need

12.1.6. Top 10 important features of positive class from SET 1

In [76]:

```
set1_features_positive_probs = []
for a in range(len(set1_features_names)):
    set1_features_positive_probs.append(naive_bayes.feature_log_prob_[1,a] )
print(len(set1_features_positive_probs))
```

```
14118
```

In [77]:

```
final_bow_features = pd.DataFrame({'feature_probability_estimates_positive' :
set1_features_positive_probs, 'set1_feature_names_positive_label': set1_features_names})
a =final_bow_features.sort_values(by = ['feature_probability_estimates_positive'], ascending =
False)

a.head(10)
```

Out[77]:

	feature_probability_estimates_positive	set1_feature_names_positive_label
10454	-2.998072	students
9496	-4.143602	school
6301	-4.502546	learning
2124	-4.525588	classroom
7322	-4.799278	not
6297	-4.851869	learn
5213	-4.871385	help
6667	-5.017677	many
7155	-5.034998	nannan
11969	-5.145539	work

using another method

In [78]:

```
neg_class_prob_sorted = naive_bayes.feature_log_prob_[0, :]
pos_class_prob_sorted = naive_bayes.feature_log_prob_[1, :]

# print(np.take(count_vect.get_feature_names(), neg_class_prob_sorted[:10]))
# print(np.take(count_vect.get_feature_names(), pos_class_prob_sorted[:10]))
```

In [79]:

```
#top 10 negative features
final_bow_features = pd.DataFrame({'feature_probability_estimates_negative' :
neg_class_prob_sorted, 'set1_feature_names_negative_label': set1_features_names})
a =final_bow_features.sort_values(by = ['feature_probability_estimates_negative'], ascending =
False)

a.head(10)
```

Out[79]:

	feature_probability_estimates_negative	set1_feature_names_negative_label
10454	-3.013000	students
9496	-4.101784	school
6301	-4.433578	learning
2124	-4.566558	classroom
7322	-4.777556	not
6297	-4.791060	learn
5213	-4.818509	help
7155	-4.986400	nannan
6667	-5.015207	many
7202	-5.130512	need

In [80]:

```
final_bow_features = pd.DataFrame({'feature_probability_estimates_positive' :
pos_class_prob_sorted, 'set1_feature_names_positive_label': set1_features_names})
a =final_bow_features.sort_values(by = ['feature_probability_estimates_positive'], ascending =
False)

a.head(10)
```

Out[80]:

	feature_probability_estimates_positive	set1_feature_names_positive_label
10454	-2.998072	students
9496	-4.143602	school
6301	-4.502546	learning
2124	-4.525588	classroom
7322	-4.799278	not
6297	-4.851869	learn
5213	-4.871385	help
6667	-5.017677	many
7155	-5.034998	nannan
11969	-5.145539	work

12.2. Applying Naive Bayes on TFIDF, SET 2

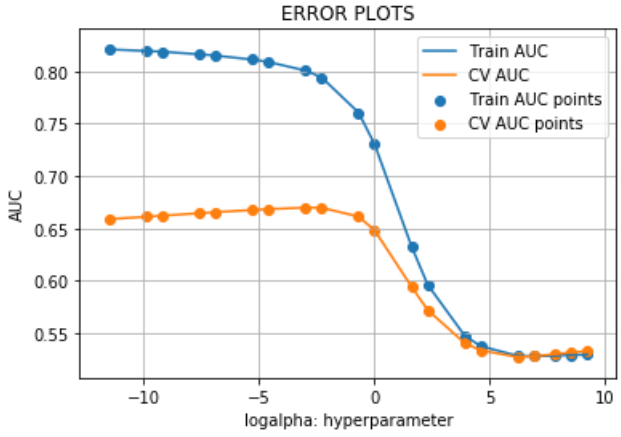
12.2.1. Finding The Best Hyperparameter "alpha"

In [98]:

```
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
"""
```

```
y_true : array, shape = [n_samples] or [n_samples, n_classes]
True binary labels or binary label indicators.

y_score : array, shape = [n_samples] or [n_samples, n_classes]
Target scores, can either be probability estimates of the positive class, confidence values, or no
n-thresholded measure of
decisions (as returned by "decision_function" on some classifiers).
For binary y_true, y_score is supposed to be the score of the class with greater label.

MultinomialNB assumes that features have multinomial distribution which is a generalization of the
binomial distribution.
Neither binomial nor multinomial distributions can contain negative values.

"""

train_auc = []
cv_auc = []
log_alpha = []

alpha = [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10, 50, 100,
         500, 1000, 2500, 5000, 10000]
for i in tqdm(alpha):
    naive_bayes = MultinomialNB(alpha = i, class_prior = [0.5,0.5], fit_prior=True)
    naive_bayes.fit(Xtrain2, y_train)

    y_train_pred_tfidf = naive_bayes.predict_proba(Xtrain2)[:,1]
    y_cv_pred_tfidf = naive_bayes.predict_proba(Xcv2)[:,1]

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred_tfidf))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred_tfidf))

for a in tqdm(alpha):
    b = math.log(a)
    log_alpha.append(b)

plt.plot(log_alpha, train_auc, label='Train AUC')
plt.plot(log_alpha, cv_auc, label='CV AUC')

plt.scatter(log_alpha, train_auc, label='Train AUC points')
plt.scatter(log_alpha, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("logalpha: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```

```
100%|██████████| 20/20 [00:03<00:00,  6.62it/s]
100%|██████████| 20/20 [00:00<?, ?it/s]
```



In [99]:

```
score_cv = [x for x in cv_auc]
optimal_alpha_cv = alpha[score_cv.index(max(score_cv))]
print("Maximum AUC score of cv is:" + ' ' + str(max(score_cv)))
print("Corresponding alpha value of cv is:",optimal_alpha_cv, '\n')
best_alpha_tfidf = optimal_alpha_cv
print(best_alpha_tfidf)
```

```
Maximum AUC score of cv is: 0.6698476595607759
Corresponding alpha value of cv is: 0.05

0.05
```

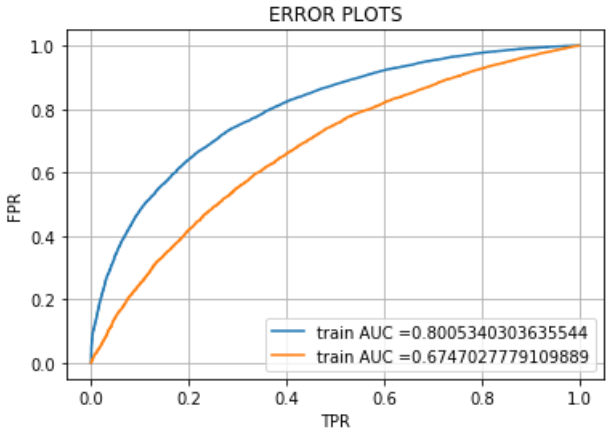
12.2.2. Testing the performance of the model on test data, plotting ROC Curves

In [100]:

```
# best_alpha_tfidf = 1.5
```

In [101]:

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


naive_bayes = MultinomialNB(alpha = best_alpha_tfidf, class_prior = [0.5,0.5], fit_prior=True)
naive_bayes.fit(Xtrain2, 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_tfidf = naive_bayes.predict_proba(Xtrain2)[:,1]
y_test_pred_tfidf = naive_bayes.predict_proba(Xtest2)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_tfidf)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_tfidf)

plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="train AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("TPR")
plt.ylabel("FPR")
plt.title("ERROR PLOTS")
plt.grid(True)
plt.show()
```



12.2.3. Building Confusion Matrix

In [102]:

```
# 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 [103]:

```
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_tfidf, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_tfidf, best_t)))
```

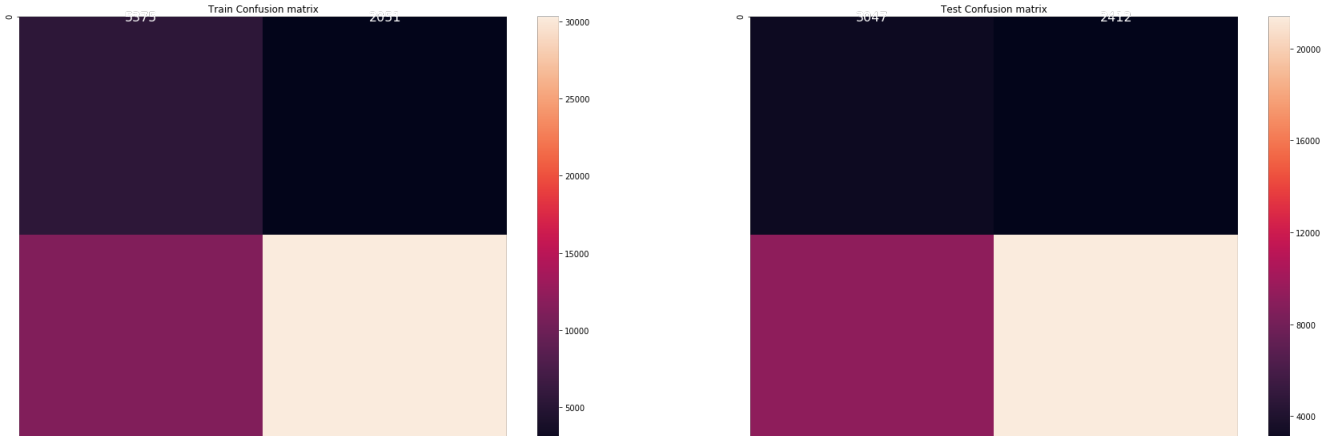
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.527076413427576 for threshold 0.494
Train confusion matrix
[[ 5375  2051]
 [11311 30304]]
Test confusion matrix
[[ 3047  2412]
 [ 9214 21379]]
```

In [104]:

```
confusion_matrix_train_tfidf = pd.DataFrame(confusion_matrix(y_train,
predict_with_best_t(y_train_pred_tfidf, best_t)))
confusion_matrix_test_tfidf = pd.DataFrame(confusion_matrix(y_test,
predict_with_best_t(y_test_pred_tfidf, best_t)))
```

In [105]:

```
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_tfidf,annot = True,annot_kws={"size": 16} ,ax = axes[0],fmt='g')
sns.heatmap(confusion_matrix_test_tfidf,annot = True ,annot_kws={"size": 16}, ax = axes[1],fmt = 'g
')
axes[0].set_title('Train Confusion matrix')
axes[1].set_title('Test Confusion matrix')
plt.show()
```



12.2.4. Appending feature names to a list

In [71]:

```
set2_features_names = []
for a in vectorizer1.get_feature_names() :# clean categories
    set2_features_names.append(a)
for a in vectorizer2.get_feature_names() :# sub categoreis
    set2_features_names.append(a)
for a in vectorizer3.get_feature_names() :# school state
    set2_features_names.append(a)
for a in vectorizer4.get_feature_names() :# teacher prefix
    set2_features_names.append(a)
for a in vectorizer5.get_feature_names() :# grade categories
    set2_features_names.append(a)

set2_features_names.append('price')
set2_features_names.append('quantity')
set2_features_names.append('teacher_number_of_previously_posted_projects')

for a in vectorizer8.get_feature_names(): # essays tfidf
    set2_features_names.append(a)
for a in vectorizer9.get_feature_names(): # titles tfidf
    set2_features_names.append(a)
print( len(set2_features_names))
```

```
14118
```

12.2.5. Top 10 important features of negative class from SET 2

In [81]:

```
#tfidf
import math
import matplotlib.pyplot as plt
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score

naive_bayes = MultinomialNB(alpha = 0.05, class_prior = [0.5,0.5], fit_prior=True)
naive_bayes.fit(Xtrain2, y_train)# fit the model
```

Out[81]:

```
MultinomialNB(alpha=0.05, class_prior=[0.5, 0.5], fit_prior=True)
```

In [82]:

```
# now make a dictionary of all the probabilities fo the weights
set2_features_negative_probs = []
for a in range(len(set2_features_names)):
    set2_features_negative_probs.append(naive_bayes.feature_log_prob_[0,a])
print(len(set2_features_negative_probs))
```

```
14118
```

In [83]:

```
#top 10 negative features
final_bow_features = pd.DataFrame({'feature_probability_estimates' : set2_features_negative_probs,
'set2_feature_names_negative_label': set2_features_names})
a =final_bow_features.sort_values(by = ['feature_probability_estimates'], ascending = False)

a.head(10)
```

Out[83]:

	feature_probability_estimates	set2_feature_names_negative_label
92	-3.469813	Mrs
4	-3.657350	Literacy_Language
98	-3.702223	Grades_PreK_2
5	-3.714696	Math_Science
93	-3.802569	Ms
95	-3.905462	Grades_3_5
28	-4.141927	Mathematics
26	-4.151857	Literacy
27	-4.470826	Literature_Writing
96	-4.616815	Grades_6_8

12.2.6. Top 10 important features of positive class from SET 2

In [84]:

```
set2_features_positive_probs = []
for a in range(len(set2_features_names)):
    set2_features_positive_probs.append(naive_bayes.feature_log_prob_[1,a] )
print(len(set2_features_positive_probs))
```

```
14118
```

In [85]:

```
#top 10 positive features
final_bow_features = pd.DataFrame({'feature_probability_estimates' : set2_features_positive_probs,
'set2_feature_names_positive_label': set2_features_names})
a =final_bow_features.sort_values(by = ['feature_probability_estimates'], ascending = False)

a.head(10)
```

Out[85]:

	feature_probability_estimates	set2_feature_names_positive_label
92	-3.437638	Mrs
4	-3.511700	Literacy_Language
98	-3.702530	Grades_PreK_2
5	-3.778541	Math_Science
93	-3.823981	Ms
95	-3.862480	Grades_3_5
26	-3.939326	Literacy
28	-4.167885	Mathematics
27	-4.385625	Literature_Writing
96	-4.674051	Grades_6_8

In [ ]:

```

```

Conclusion

In [ ]:

```

```

In [113]:

```
# 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(["BOW", "Multinomial NB", 0.5 , 0.7980,0.7061])
x.add_row(["TFIDF", "Multinomial NB", 0.05 , 0.8005,0.6747])
```

In [114]:

```
print(x)
```

```
+------------+----------------+-----------------+-----------+----------+
| Vectorizer |     Model      | Hyper parameter | Train AUC | Test AUC |
+------------+----------------+-----------------+-----------+----------+
|    BOW     | Multinomial NB |       0.5       |   0.798   |  0.7061  |
|   TFIDF    | Multinomial NB |       0.05      |   0.8005  |  0.6747  |
+------------+----------------+-----------------+-----------+----------+
```

So, we conclude by the above observations that Encoding our text data using Bag of Words and applying Multinomial Naive Bayes give us more Accuracy on Test data

In [ ]:

In [ ]: