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 = 50000)
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 (50000, 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]:

```
# we get the cost of the project using resource.csv file
resource_data.head(2)
resource_data.loc[resource_data['id'] == 'p000001']
```

Out[7]:

	id	description	quantity	price
414179	p000001	Cap Barbell 300 Pound Olympic Set, Grey	2	261.08
414180	p000001	Cap Barbell Power Rack Exercise Stand	2	89.00
414181	p000001	Marcy SB-10510 Flat Bench	1	85.49
414182	p000001	ProSource Puzzle Exercise Mat High Quality EVA	2	23.99

id	description	quantity	price

In [8]:

```
resource_data.iloc[414179]
```

Out[8]:

```
id                                                p000001
description    Cap Barbell 300 Pound Olympic Set, Grey
quantity                                                2
price                                              261.08
Name: 414179, dtype: object
```

In [9]:

```
# 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[9]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [10]:

```
price_data.shape
```

Out[10]:

```
(260115, 3)
```

In [11]:

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

In [12]:

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

Out[12]:

	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 [13]:

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

3. Text Preprocessing

3.1. Concatenating all essay text

In [14]:

```
# 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 [15]:

```
# 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 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 [16]:

```
# 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 [17]:

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

```
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 i
n Mind and work hard everyday to reach our goals.\r\n\r\nWe do not 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-incom
e/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 would not know the daily struggle f
or some of them.  I ask you. How can you learn with your belly growling?  How can I provide the ab
solute 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 t
he fire in young minds.Receiving books written by the same author will teach students how to
develop their own Writer is 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 lo
ve 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 is craft, go
on adventures in their minds, and develop a tenacious love for reading for the sake of
reading.nannan
==================================================
```

In [18]:

```
# \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)
```

```
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 i
n Mind and work hard everyday to reach our goals.    We do not believe in making excuses, but ther
e 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 ou
t of the classroom.  Even though, as a visitor to my classroom, you would not know the daily
struggle for some of them.  I ask you. How can you learn with your belly growling?  How can I prov
ide the absolute best learning environment when we do not have the money to buy research-based mat
erials?      "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 ligh
t the fire in young minds.Receiving books written by the same author will teach students how to de
velop their own Writer is Craft.  It will inspire them to think about different ways established a
uthors have developed successful text that appeal to various audiences.      We never forget our fi
rst love.  My mother read the Berenstain Bears series to me when I was five and I fell in love wit
h 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, C
urious George!  Thank you Margaret and H.A. Rey for creating a series that captured my heart and a
ttention.      As 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 is craft, go on adventures
in their minds, and develop a tenacious love for reading for the sake of reading.nannan
```

In [19]:

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

```
We have GRIT If you want to meet tenacious respectful seven year olds with growth mindsets you nee
d to come to our classroom We give hugs high fives and compliments We Begin with the End in Mind a
nd work hard everyday to reach our goals We do not believe in making excuses but there are times i
n life when you just need to ask for help As a classroom teacher in a low income high poverty scho
ol district my 2nd grade students face real life struggles both in and out of the classroom Even t
hough as a visitor to my classroom you would not know the daily struggle for some of them I ask yo
u How can you learn with your belly growling How can I provide the absolute best learning environm
ent when we do not have the money to buy research based materials 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 is Craft It will inspire them to think about different ways established authors have developed successful text that appeal to various audiences We 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 As 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 is craft go on adventures in their minds and develop a tenacious love for reading for the sake of reading n annan

In [20]:

```
# 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 [21]:

```
# 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%|████████████████████████████████████████████████████████████████| 50000/50000
[00:40<00:00, 1222.87it/s]
```

In [22]:

```
# after preprocesing
preprocessed_essays[49999]
```

Out[22]:

'grit want meet tenacious respectful seven year olds growth mindsets need come classroom give hugs

```
high fives compliments begin end mind work hard everyday reach goals not believe making excuses ti
mes life need ask help classroom teacher low income high poverty school district 2nd grade student
s face real life struggles classroom even though visitor classroom would not know daily struggle a
sk learn belly growling provide absolute best learning environment not money buy research based ma
terials education not filling pail lighting fire william butler yeats not asking fill pail things
help provide resources light fire young minds receiving books written author teach students
develop writer craft inspire think different ways established authors developed successful text ap
peal various audiences never forget first love mother read berenstain bears series five fell love
berenstain family took public library every week would hunt books written stan jan berenstain next
curious monkey man yellow hat curious george thank margaret h rey creating series captured heart a
ttention teacher hope dream inspire students classroom find first love reading help help discover
writer craft go adventures minds develop tenacious love reading sake reading nannan'
```

In [23]:

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

Out[23]:

	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

3.3. Preprocessing Title text

In [24]:

```
# 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 [25]:

```
# 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
```

```
    # 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 [26]:

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

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

In [27]:

```
# \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 [28]:

```
#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 [29]:

```
# 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', 'each', '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', "doesn'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 [30]:

```
# 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%|██████████████████████████████████████████████████████████████████████████| 50000/50000
[00:01<00:00, 25977.01it/s]
```

In [31]:

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

Out[31]:

```
'need move input'
```

In [32]:

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

Out[32]:

	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 [33]:

```
project_data.head()
```

Out[33]:

	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

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_grade_cate
4	172407	p104768	be1f7507a41f8479dc06f047086a39ec	Mrs.	TX	2016-07-11 01:10:09	Grades P

4. Preprocesseing of Categorical Data

4.1. Preprocessing project_grade_category

In [34]:

```
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 [35]:

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

Out[35]:

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

In [36]:

```
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[36]:

	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 [37]:

```
catogories = list(project_data['project_subject_categories'].values)
# print(catogories)
# 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 [38]:

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

Out[38]:

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

In [ ]:

```

```

In [39]:

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

Out[39]:

	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 [40]:

```
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 [41]:

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

Out[41]:

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

In [42]:

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

Out[42]:

	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 [43]:

```
project_data.head()
```

Out[43]:

	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...

In [ ]:

```

```

In [ ]:

```

```

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

In [44]:

```
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 [45]:

```
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 [46]:

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

```
(22445, 19)
(16500, 19)
(11055, 19)
```

In [47]:

```
X_train.head()
```

Out[47]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_state	project_submitted_datetime	project_essay
3769	109911	p001979	7cfc096cdcb5be8dcab0debc93450b1f	Ms.	CA	2017-03-29 13:21:09	I have wonderful gro of energe transi
5753	173547	p207143	78563a0f693b77d5cad7216898d5b695	Ms.	GA	2017-01-10 20:37:07	Every day is adventure, a every day is a
42725	78155	p166218	14d6ac51da124bb70c40996becdd1be3	Mrs.	MI	2016-09-12 16:28:32	My Young Fiv are full of ene and love t
48655	1330	p210703	1c3a3c0a8d76ffa9eaf6500c142b19cd	Ms.	SC	2017-04-24 18:43:59	Our libr serves as a h of learni conne
26811	177051	p257299	fdbf9cc3c461c85de0e357431673f2f3	Ms.	WI	2016-08-19 11:54:11	My students a a great bunch kids! They a

In [48]:

```
y_train.head(10)
```

Out[48]:

```
3769     1
5753     1
42725    1
48655    1
26811    1
22326    1
9733     1
24496    1
```

```
2288     1
44492    1
Name: project_is_approved, dtype: int64
```

7. Encoding Categorical Data

7.1. One Hot Encoding of clean_categories

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['clean_categories'].values:
#     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 [50]:

```
# 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(X_train['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  (22445, 9)
Shape of matrix after one hot encodig  (16500, 9)
Shape of matrix after one hot encodig  (11055, 9)
```

7.2. One Hot Encoding of clean_subcategories

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['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 [52]:

```
# 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  (22445, 30)
Shape of matrix after one hot encodig  (16500, 30)
Shape of matrix after one hot encodig  (11055, 30)
```

7.3. One Hot Encoding of school_state

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['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 [54]:

```
# 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  (22445, 51)
Shape of matrix after one hot encoding  (16500, 51)
Shape of matrix after one hot encoding  (11055, 51)
```

7.4. One Hot Encoding of teacher_prefix

In [55]:

```
# # 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 [56]:

```
# 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  (22445, 5)
Shape of matrix after one hot encoding  (16500, 5)
Shape of matrix after one hot encoding  (11055, 5)
```

7.5. One Hot Encoding of project_grade_clean_category

In [57]:

```
# # 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 [58]:

```
# 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  (22445, 4)
Shape of matrix after one hot encoding  (16500, 4)
Shape of matrix after one hot encoding  (11055, 4)
```

8. Encoding of Text Data

8.1. BOW encoding of preprocessed_essays

In [59]:

```
# 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  (22445, 8780)
Shape of matrix after one hot encodig  (16500, 8780)
Shape of matrix after one hot encodig  (11055, 8780)
```

8.2. BOW encoding of preprocessed_titles

In [60]:

```
# 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  (22445, 1154)
Shape of matrix after one hot encodig  (16500, 1154)
Shape of matrix after one hot encodig  (11055, 1154)
```

8.3. TFIDF encoding of preprocessed_essays

In [61]:

```
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  (22445, 8780)
Shape of matrix after one hot encodig  (16500, 8780)
Shape of matrix after one hot encodig  (11055, 8780)
```

8.4. TFIDF encoding of preprocessed_titles

In [62]:

```
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  (22445, 1154)
Shape of matrix after one hot encodig  (16500, 1154)
Shape of matrix after one hot encodig  (11055, 1154)
```

8.5. Average Word2Vec encoding of preprocessed_essays on

Train Data

In [63]:

```
# stronging variables into pickle files python: http://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-python/
# make sure you have the glove_vectors file
with open('D:\glove_vectors', 'rb') as f:
    model = pickle.load(f)
    glove_words =  set(model.keys())
```

In [64]:

```
# 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(300) # 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 glove_words:
            vector += model[word]
            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%|██████████████████████████████████████████████████████████████████████████| 22445/22445
[00:10<00:00, 2088.86it/s]
```

```
22445
300
```

In [65]:

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

```
(22445, 300)
```

8.5.1Average Word2Vec encoding of preprocessed_essays on Test Data

In [66]:

```
# 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(300) # 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 glove_words:
            vector += model[word]
            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%|██████████████████████████████████████████████████████████████████████████| 16500/16500
[00:07<00:00, 2210.28it/s]
```

```
16500
300
```

In [67]:

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

```
(16500, 300)
```

8.5.2. Average Word2Vec encoding of preprocessed_essays on CV Data

In [68]:

```
# 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
    vector = np.zeros(300) # 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 glove_words:
            vector += model[word]
            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%|████████████████████████████████████████████████████████████████████████████| 11055/11055
[00:04<00:00, 2293.59it/s]

11055
300
```

In [69]:

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

```
(11055, 300)
```

8.6. Average Word2Vec encoding of preprocessed_titles on Train Data

In [70]:

```
#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(300) # 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 glove_words:
            vector += model[word]
            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%|████████████████████████████████████████████████████████████████| 22445/22445
[00:00<00:00, 37953.29it/s]
```

```
22445
300
```

In [71]:

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

```
(22445, 300)
```

8.6.1. Average Word2Vec encoding of preprocessed_titles on Test Data

In [72]:

```
#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(300) # 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 glove_words:
            vector += model[word]
            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%|████████████████████████████████████████████████████████████████| 16500/16500
[00:00<00:00, 35597.57it/s]
```

```
16500
300
```

In [73]:

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

```
(16500, 300)
```

8.6.2. Average Word2Vec encoding of preprocessed_titles on CV Data

In [74]:

```
#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(300) # 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 glove_words:
            vector += model[word]
            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%|██████████████████████████████████████████████████████████████████████████| 11055/11055
[00:00<00:00, 29724.96it/s]
```

```
11055
300
```

In [75]:

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

```
(11055, 300)
```

8.7. TFIDF weighted Word2Vec encoding of preprocessed_essays on Train Data

In [76]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_essays'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [77]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_weighted_w2v_vectors_eassays_Xtrain = []; # the avg-w2v for each sentence/review is stored i
n this list
for sentence in tqdm(X_train['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =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 glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_weighted_w2v_vectors_eassays_Xtrain.append(vector)

print(len(tfidf_weighted_w2v_vectors_eassays_Xtrain))
print(len(tfidf_weighted_w2v_vectors_eassays_Xtrain[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 22445/22445 [01:
04<00:00, 346.72it/s]
```

```
22445
300
```

In [78]:

```
tfidf_weighted_w2v_on_essay_matrix_Xtrain = np.vstack(tfidf_weighted_w2v_vectors_eassays_Xtrain)
print(tfidf_weighted_w2v_on_essay_matrix_Xtrain.shape)
```

```
(22445, 300)
```

8.7.1. TFIDF weighted Word2Vec encoding of preprocessed_essays on Test Data

In [79]:

```
# # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
# tfidf_model = TfidfVectorizer()
# tfidf_model.fit(X_test['preprocessed_essays'].values)
# # we are converting a dictionary with word as a key, and the idf as a value
# dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
# tfidf_words = set(tfidf_model.get_feature_names())
```

In [80]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_weighted_w2v_vectors_eassays_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(300) # as word vectors are of zero length
    tf_idf_weight =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 glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_weighted_w2v_vectors_eassays_Xtest.append(vector)

print(len(tfidf_weighted_w2v_vectors_eassays_Xtest))
print(len(tfidf_weighted_w2v_vectors_eassays_Xtest[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 16500/16500 [00:
48<00:00, 339.60it/s]
```

```
16500
300
```

In [81]:

```
tfidf_weighted_w2v_on_essay_matrix_Xtest = np.vstack(tfidf_weighted_w2v_vectors_eassays_Xtest)
print(tfidf_weighted_w2v_on_essay_matrix_Xtest.shape)
```

```
(16500, 300)
```

8.7.2. TFIDF weighted Word2Vec encoding of preprocessed_essays on CV Data

In [82]:

```
# # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
# tfidf_model = TfidfVectorizer()
# tfidf_model.fit(X_cv['preprocessed_essays'].values)
# # we are converting a dictionary with word as a key, and the idf as a value
# dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
```

```
# tfidf_words = set(tfidf_model.get_feature_names())
```

In [83]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_weighted_w2v_vectors_eassays_Xcv = []; # the avg-w2v for each sentence/review is stored in t
his list
for sentence in tqdm(X_cv['preprocessed_essays'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =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 glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(sentence.count(word)/len(sentence.split())) # getting the tf
idf value for each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_weighted_w2v_vectors_eassays_Xcv.append(vector)

print(len(tfidf_weighted_w2v_vectors_eassays_Xcv))
print(len(tfidf_weighted_w2v_vectors_eassays_Xcv[0]))
```

```
100%|████████████████████████████████████████████████████████████████████████████| 11055/11055 [00:
31<00:00, 346.11it/s]
```

```
11055
300
```

In [84]:

```
tfidf_weighted_w2v_on_essay_matrix_Xcv = np.vstack(tfidf_weighted_w2v_vectors_eassays_Xcv)
print(tfidf_weighted_w2v_on_essay_matrix_Xcv.shape)
```

```
(11055, 300)
```

8.8. TFIDF Weighted Word2Vec encoding of preprocessed_titles on Train Data

In [85]:

```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
tfidf_model = TfidfVectorizer()
tfidf_model.fit(X_train['preprocessed_titles'].values)
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [86]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_weighted_w2v_vectors_title_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(300) # as word vectors are of zero length
    tf_idf_weight =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 glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(t.count(word)/len(t.split())) # getting the tfidf value for
each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
```

```
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_weighted_w2v_vectors_title_Xtrain.append(vector)

print(len(tfidf_weighted_w2v_vectors_title_Xtrain))
print(len(tfidf_weighted_w2v_vectors_title_Xtrain[0]))
```

```
100%|████████████████████████████████████████████████████████████████| 22445/22445
[00:01<00:00, 20520.66it/s]
```

```
22445
300
```

In [87]:

```
tfidf_weighted_w2v_on_title_matrix_Xtrain = np.vstack(tfidf_weighted_w2v_vectors_title_Xtrain)
print(tfidf_weighted_w2v_on_title_matrix_Xtrain.shape)
```

```
(22445, 300)
```

8.8.1.TFIDF Weighted Word2Vec encoding of preprocessed_titles on Test Data

In [88]:

```
# # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
# tfidf_model = TfidfVectorizer()
# tfidf_model.fit(X_test['preprocessed_titles'].values)
# # we are converting a dictionary with word as a key, and the idf as a value
# dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
# tfidf_words = set(tfidf_model.get_feature_names())
```

In [89]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_weighted_w2v_vectors_title_Xtest = []; # the avg-w2v for each sentence/review is stored in t
his list
for t in tqdm(X_test['preprocessed_titles'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =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 glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(t.count(word)/len(t.split())) # getting the tfidf value for
each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_weighted_w2v_vectors_title_Xtest.append(vector)

print(len(tfidf_weighted_w2v_vectors_title_Xtest))
print(len(tfidf_weighted_w2v_vectors_title_Xtest[0]))
```

```
100%|████████████████████████████████████████████████████████████████| 16500/16500
[00:00<00:00, 20521.58it/s]
```

```
16500
300
```

In [90]:

```
tfidf_weighted_w2v_on_title_matrix_Xtest = np.vstack(tfidf_weighted_w2v_vectors_title_Xtest)
```

```
print(tfidf_weighted_w2v_on_title_matrix_Xtest.shape)
```

```
(16500, 300)
```

8.8.2. TFIDF Weighted Word2Vec encoding of preprocessed_titles on CV Data

In [91]:

```
# # S = ["abc def pqr", "def def def abc", "pqr pqr def"]
# tfidf_model = TfidfVectorizer()
# tfidf_model.fit(X_cv['preprocessed_titles'].values)
# # we are converting a dictionary with word as a key, and the idf as a value
# dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
# tfidf_words = set(tfidf_model.get_feature_names())
```

In [92]:

```
# average Word2Vec
# compute average word2vec for each review.
tfidf_weighted_w2v_vectors_title_Xcv = []; # the avg-w2v for each sentence/review is stored in thi
s list
for t in tqdm(X_cv['preprocessed_titles'].values): # for each review/sentence
    vector = np.zeros(300) # as word vectors are of zero length
    tf_idf_weight =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 glove_words) and (word in tfidf_words):
            vec = model[word] # getting the vector for each word
            # here we are multiplying idf value(dictionary[word]) and the tf
value((sentence.count(word)/len(sentence.split())))
            tf_idf = dictionary[word]*(t.count(word)/len(t.split())) # getting the tfidf value for
each word
            vector += (vec * tf_idf) # calculating tfidf weighted w2v
            tf_idf_weight += tf_idf
    if tf_idf_weight != 0:
        vector /= tf_idf_weight
    tfidf_weighted_w2v_vectors_title_Xcv.append(vector)

print(len(tfidf_weighted_w2v_vectors_title_Xcv))
print(len(tfidf_weighted_w2v_vectors_title_Xcv[0]))
```

```
100%|██████████████████████████████████████████████████████████████████████████| 11055/11055
[00:00<00:00, 17870.84it/s]
```

```
11055
300
```

In [93]:

```
tfidf_weighted_w2v_on_title_matrix_Xcv = np.vstack(tfidf_weighted_w2v_vectors_title_Xcv)
print(tfidf_weighted_w2v_on_title_matrix_Xcv.shape)
```

```
(11055, 300)
```

9. Encoding of Numerical Data

We Encode our data using MinMaxScalar() because it produces positive values and Chi square test can be applied on it

9.1.1. Encoding of price on Train Data

In [94]:

```
# check this one: https://www.youtube.com/watch?v=0HOqOcln3Z4&t=530s
```

```
# 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 [95]:

```
price_standardized_Xtrain
```

Out[95]:

```
array([[0.00862764],
       [0.02414359],
       [0.02759224],
       ...,
       [0.02881147],
       [0.14654984],
       [0.01988978]])
```

In [96]:

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

```
(22445, 1)
(16500, 1)
(11055, 1)
```

9.2.1. Encoding of quantity on Train Data

In [97]:

```
# 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 [98]:

```
quantity_standardized_Xtrain
```

Out[98]:

```
array([[0.01223582],
       [0.        ],
       [0.01223582],
       ...,
       [0.        ],
       [0.01446051],
       [0.01112347]])
```

In [99]:

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

```
(22445, 1)
(16500, 1)
(11055, 1)
```

9.3.1. Encoding of teacher_number_of_previously_posted_projects on Train Data

In [100]:

```
# 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 [101]:

```
teacher_number_of_previously_posted_projects_standardized_Xtrain
```

Out[101]:

```
array([[0.0212766 ],
       [0.0141844 ],
       [0.00472813],
       ...,
       [0.0141844 ],
       [0.        ],
       [0.00236407]])
```

In [102]:

```
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)
```

```
(22445, 1)
(16500, 1)
(11055, 1)
```

10. Printing Dimensions of all Preprocessed Data

In [103]:

```
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(average_w2v_on_essay_Xtrain.shape)
print(average_w2v_on_essay_Xtest.shape)
print(average_w2v_on_essay_Xcv.shape)
print(average_w2v_on_titles_Xtrain.shape)
print(average_w2v_on_titles_Xtest.shape)
print(average_w2v_on_titles_Xcv.shape)
print(tfidf_weighted_w2v_on_essay_matrix_Xtrain.shape)
print(tfidf_weighted_w2v_on_essay_matrix_Xtest.shape)
print(tfidf_weighted_w2v_on_essay_matrix_Xcv.shape)
print(tfidf_weighted_w2v_on_title_matrix_Xtrain.shape)
print(tfidf_weighted_w2v_on_title_matrix_Xtest.shape)
print(tfidf_weighted_w2v_on_title_matrix_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)
```

```
(22445, 9)
(16500, 9)
(11055, 9)
(22445, 30)
(16500, 30)
(11055, 30)
(22445, 51)
(16500, 51)
(11055, 51)
(22445, 5)
(16500, 5)
(11055, 5)
(22445, 4)
(16500, 4)
(11055, 4)
(22445, 8780)
(16500, 8780)
(11055, 8780)
(22445, 1154)
(16500, 1154)
(11055, 1154)
(22445, 8780)
(16500, 8780)
(11055, 8780)
(22445, 1154)
(16500, 1154)
(11055, 1154)
(22445, 300)
(16500, 300)
(11055, 300)
(22445, 300)
(16500, 300)
(11055, 300)
(22445, 300)
(16500, 300)
(11055, 300)
(22445, 300)
(16500, 300)
(11055, 300)
(22445, 1)
(16500, 1)
(11055, 1)
(22445, 1)
```

```
(22445, 1)
(16500, 1)
(11055, 1)
(22445, 1)
(16500, 1)
(11055, 1)
```

11. Creating Different Sets of Data for Training Model

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

In [104]:

```
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)
```

```
(22445, 10036) (22445,)
(16500, 10036) (16500,)
(11055, 10036) (11055,)
```

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

In [105]:

```
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)
```

```
(22445, 10036) (22445,)
(16500, 10036) (16500,)
(11055, 10036) (11055,)
```

Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)

In [106]:

```
Xtrain3 =
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,average_w2v_on_essay_Xtrain,aver
age_w2v_on_titles_Xtrain)).tocsr()
Xtest3 = 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,average_w2v_on_essay_Xtest,aver
age_w2v_on_titles_Xtest)).tocsr()
Xcv3 =
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,average_w2v_on_essay_Xcv,average_w2v_on_titles_Xcv)).tocs
r()

print(Xtrain3.shape,y_train.shape)
print(Xtest3.shape,y_test.shape)
print(Xcv3.shape,y_cv.shape)
```

```
(22445, 702) (22445,)
(16500, 702) (16500,)
(11055, 702) (11055,)
```

Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_eassay (TFIDF W2V)

In [107]:

```
from scipy.sparse import hstack
# with the same hstack function we are concatinating a sparse matrix and a dense matirx :)
Xtrain4 =
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,tfidf_weighted_w2v_on_essay_matri
_Xtrain,tfidf_weighted_w2v_on_title_matrix_Xtrain)).tocsr()
Xtest4 = 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,tfidf_weighted_w2v_on_essay_matr
x_Xtest,tfidf_weighted_w2v_on_title_matrix_Xtest)).tocsr()
Xcv4 =
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,tfidf_weighted_w2v_on_essay_matrix_Xcv,tfidf_weighted_w2v_
on_title_matrix_Xcv)).tocsr()

print(Xtrain4.shape,y_train.shape)
print(Xtest4.shape,y_test.shape)
print(Xcv4.shape,y_cv.shape)
```

```
(22445, 702) (22445,)
(16500, 702) (16500,)
(11055, 702) (11055,)
```

12. Appling KNN on different kind of featurization

12.1. Applying KNN brute force on BOW, SET 1

Function for predicting Target values Batchwise

In [108]:

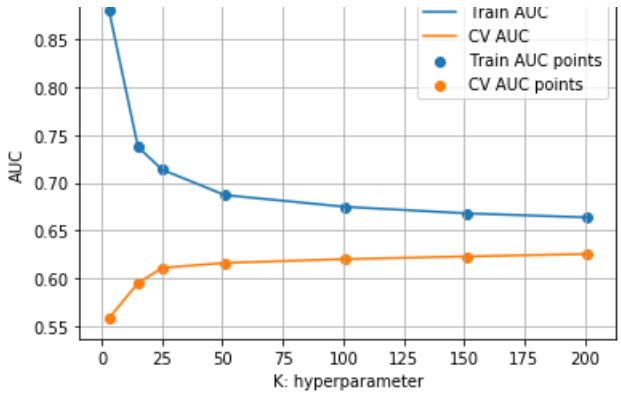
```
def batch_predict(clf, data):
    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of the posi
tive class
    # not the predicted outputs

    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 "K"

In [109]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
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 = []
K = [3, 15, 25, 51, 101,151,201]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(Xtrain1, y_train)

    y_train_pred_bow = batch_predict(neigh, Xtrain1)
    y_cv_pred_bow = batch_predict(neigh, Xcv1)

    # 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))

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

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

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

```
100%|██████████████████████████████████████████████████████████████████████████████████████████| 7/7 [09
:52<00:00, 85.65s/it]
```



In [110]:

```
score_cv = [x for x in cv_auc]
optimal_K_cv = K[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_K_cv, '\n')
best_K_bow = optimal_K_cv
print(best_K_bow)
```

```
Maximum AUC score of cv is: 0.6252560415261812
Corresponding alpha value of cv is: 201

201
```

In [111]:

```
# # https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
# from sklearn.model_selection import GridSearchCV
# from scipy.stats import randint as sp_randint
# from sklearn.model_selection import RandomizedSearchCV

# neigh = KNeighborsClassifier(n_jobs=-1)
# parameters = {'n_neighbors':sp_randint(50, 100)}
# clf = RandomizedSearchCV(neigh, parameters, cv=3, scoring='roc_auc')
# clf.fit(X_tr, y_train)

# results = pd.DataFrame.from_dict(clf.cv_results_)
# results = results.sort_values(['param_n_neighbors'])

# train_auc= results['mean_train_score']
# train_auc_std= results['std_train_score']
# cv_auc = results['mean_test_score']
# cv_auc_std= results['std_test_score']
# K =  results['param_n_neighbors']

# plt.plot(K, train_auc, label='Train AUC')
# # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# # plt.gca().fill_between(K, train_auc - train_auc_std,train_auc +
train_auc_std,alpha=0.2,color='darkblue')

# plt.plot(K, cv_auc, label='CV AUC')
# # this code is copied from here: https://stackoverflow.com/a/48803361/4084039
# # plt.gca().fill_between(K, cv_auc - cv_auc_std,cv_auc +
cv_auc_std,alpha=0.2,color='darkorange')

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


# plt.legend()
# plt.xlabel("K: hyperparameter")
# plt.ylabel("AUC")
# plt.title("Hyper parameter Vs AUC plot")
# plt.grid()
# plt.show()

# results.head()
```

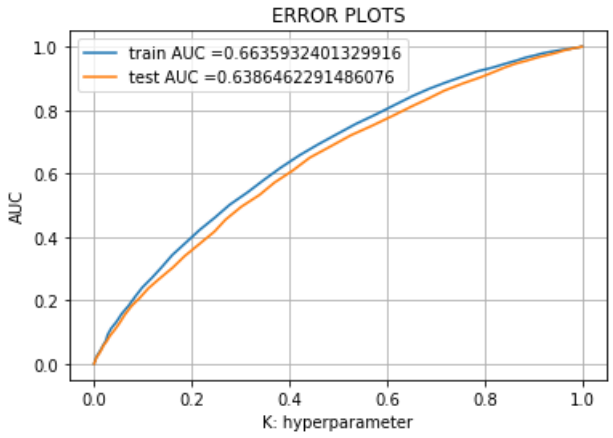
12.1.2. Testing the performance of the model on test data, plotting ROC Curves

In [112]:

```
# # from the error plot we choose K such that, we will have maximum AUC on cv data and gap between
the train and cv is less
# # Note: based on the method you use you might get different hyperparameter values as best one
# # so, you choose according to the method you choose, you use gridsearch if you are having more c
omputing power and note it will take more time
# # if you increase the cv values in the GridSearchCV you will get more rebust results.

# #here we are choosing the best_k based on forloop results
# best_k = 101
```

In [117]:

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


neigh = KNeighborsClassifier(n_neighbors=best_K_bow, n_jobs=-1)
neigh.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 = batch_predict(neigh, Xtrain1)
y_test_pred_bow = batch_predict(neigh, Xtest1)

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("K: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS")
plt.grid()
plt.show()
```



12.1.3. Building Confusion Matrix

In [118]:

```
# 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 [119]:

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

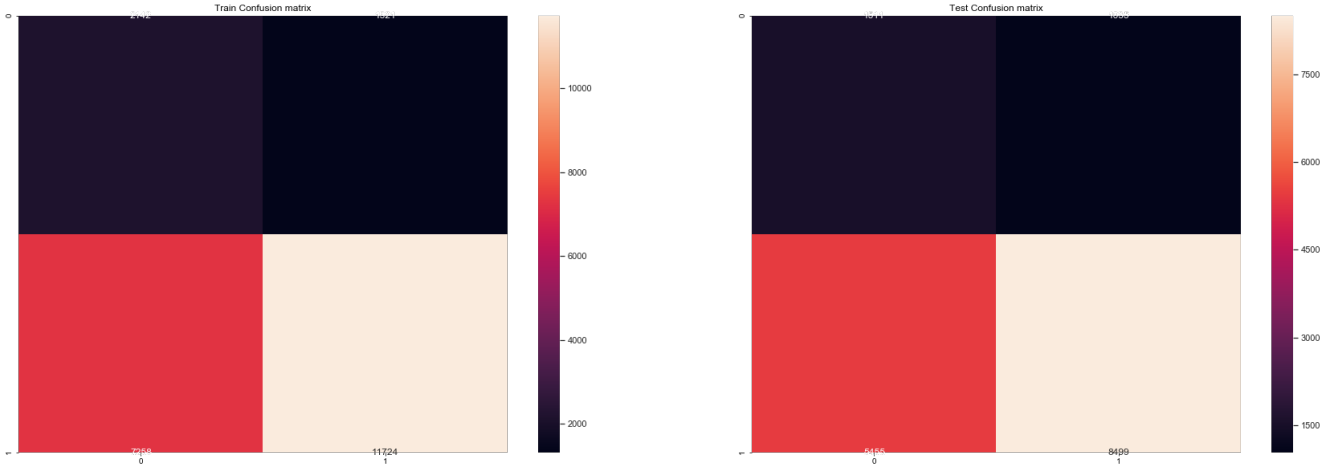
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.38203294438280105 for threshold 0.791
Train confusion matrix
[[ 2142  1321]
 [ 7258 11724]]
Test confusion matrix
[[1511 1035]
 [5455 8499]]
```

In [120]:

```
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 [121]:

```
import seaborn as sns
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(30,10))
sns.set()
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()
```



12.2. Applying KNN brute force on TFIDF, SET 2

12.2.1. Finding The Best Hyperparameter "K"

In [126]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
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 = []
K = [3, 15, 25, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(Xtrain2, y_train)

    y_train_pred_tfidf = batch_predict(neigh, Xtrain2)
    y_cv_pred_tfidf = batch_predict(neigh, Xcv2)

    # 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))

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

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

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

```
100%|██████████| 5/5 [06
:03<00:00, 71.76s/it]
```

In [127]:

```
score_cv = [x for x in cv_auc]
optimal_K_cv = K[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_K_cv, '\n')
best_K_tfidf = optimal_K_cv
print(best_K_tfidf)
```

```
Maximum AUC score of cv is: 0.5832065488418775
Corresponding alpha value of cv is: 101

101
```

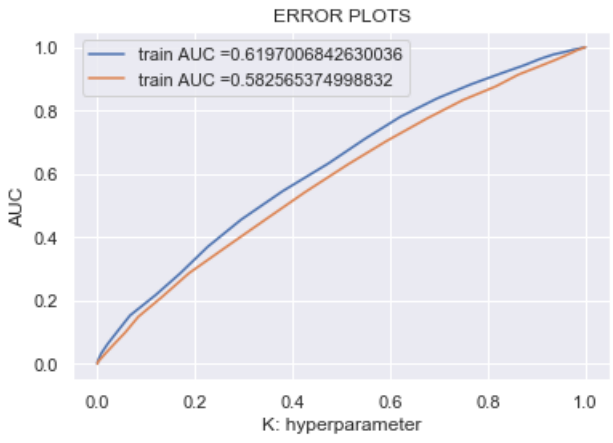
12.2.2. Testing the performance of the model on test data, plotting ROC Curves

In [128]:

```
# best_K_tfidf = 101
```

In [129]:

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


neigh = KNeighborsClassifier(n_neighbors=best_K_tfidf, n_jobs=-1)
neigh.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 = batch_predict(neigh, Xtrain2)
y_test_pred_tfidf = batch_predict(neigh, Xtest2)

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



12.2.3. Building Confusion Matrix

In [130]:

```
# 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 [143]:

```
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)))
```

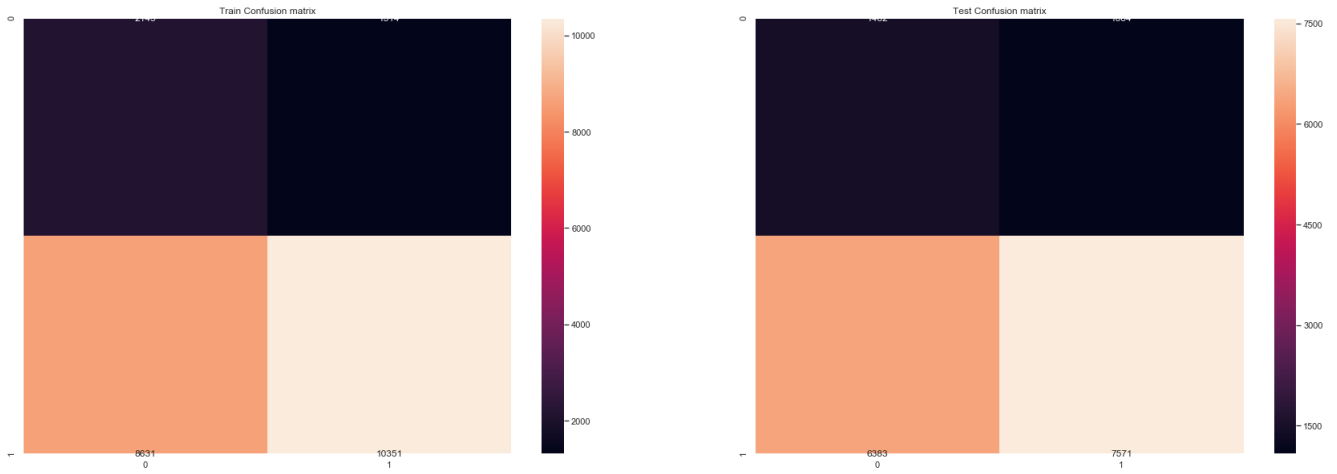
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.36861075706994545 for threshold 0.851
Train confusion matrix
[[ 2149  1314]
 [ 8631 10351]]
Test confusion matrix
[[1462 1084]
 [6383 7571]]
```

In [144]:

```
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 [145]:

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



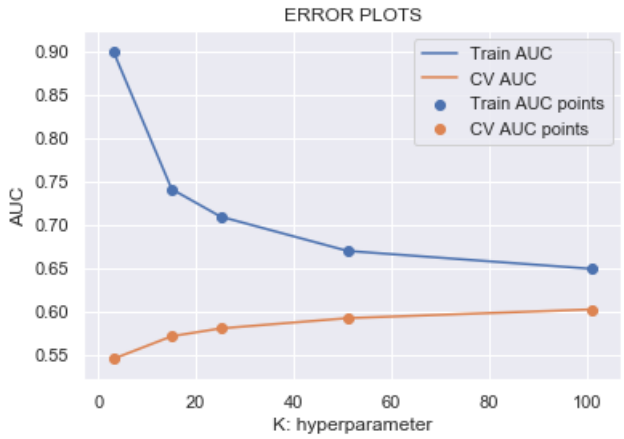
12.3. Applying KNN brute force on AVG W2V, SET 3

12.3.1. Finding The Best Hyperparameter "K"

In [134]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
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 = []
K = [3, 15, 25, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(Xtrain3, y_train)

    y_train_pred_avgw2v = batch_predict(neigh, Xtrain3)
    y_cv_pred_avgw2v = batch_predict(neigh, Xcv3)

    # 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_avgw2v))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred_avgw2v))

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

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

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

```
100%|██████████| 5/5 [1:01:07<00:00, 733.02s/it]
```



In [135]:

```
score_cv = [x for x in cv_auc]
optimal_K_cv = K[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_K_cv, '\n')
best_K_avgw2v = optimal_K_cv
print(best_K_avgw2v)
```

```
Maximum AUC score of cv is: 0.602242445151881
Corresponding alpha value of cv is: 101

101
```

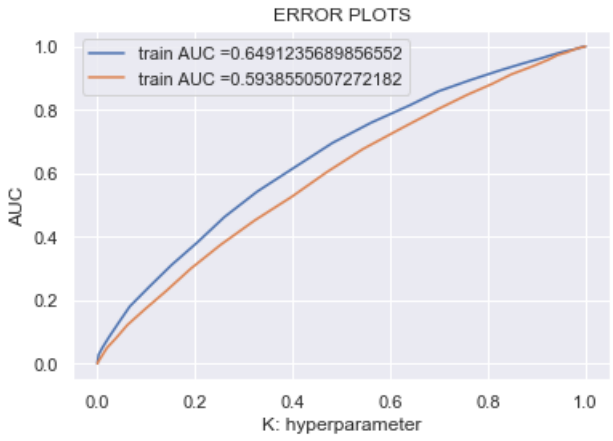
12.3.2. Testing the performance of the model on test data, plotting ROC Curves

In [136]:

```
# best_k_tfidf = 101
```

In [138]:

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


neigh = KNeighborsClassifier(n_neighbors=best_K_avgw2v, n_jobs=-1)
neigh.fit(Xtrain3, 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_avgw2v = batch_predict(neigh, Xtrain3)
y_test_pred_avgw2v = batch_predict(neigh, Xtest3)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_avgw2v)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_avgw2v)

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



12.3.3. Building Confusion Matrix

In [139]:

```
# 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 [146]:

```
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_avgw2v, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_avgw2v, best_t)))
```

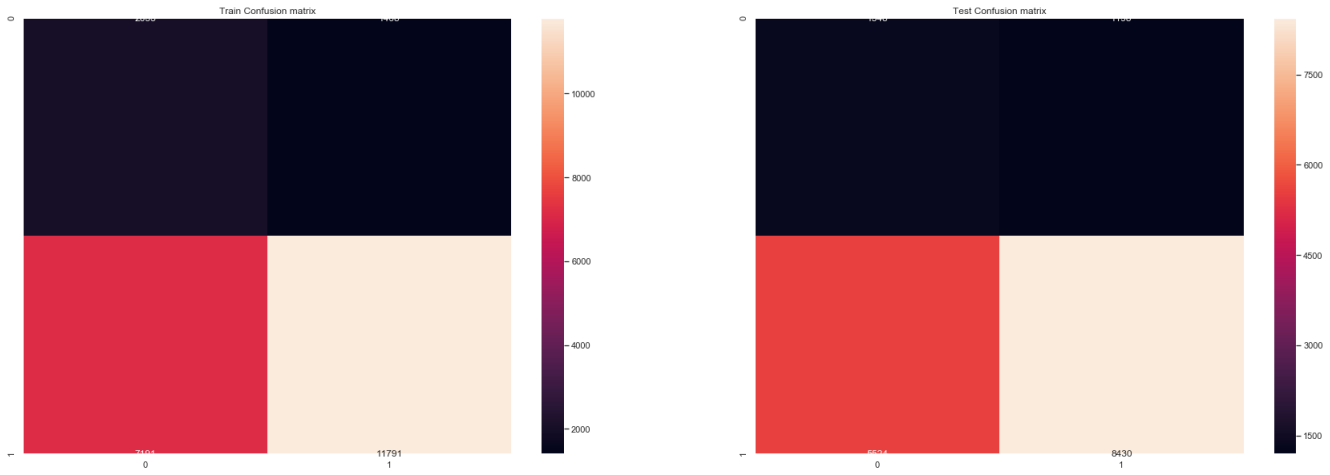
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.36861075706994545 for threshold 0.851
Train confusion matrix
[[ 2055  1408]
 [ 7191 11791]]
Test confusion matrix
[[1348 1198]
 [5524 8430]]
```

In [147]:

```
confusion_matrix_train_avgw2v = pd.DataFrame(confusion_matrix(y_train,
predict_with_best_t(y_train_pred_avgw2v, best_t)))
confusion_matrix_test_avgw2v = pd.DataFrame(confusion_matrix(y_test,
predict_with_best_t(y_test_pred_avgw2v, best_t)))
```

In [148]:

```
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_avgw2v,annot = True ,ax = axes[0],fmt='g')
sns.heatmap(confusion_matrix_test_avgw2v,annot = True , ax = axes[1],fmt = 'g')
axes[0].set_title('Train Confusion matrix')
axes[1].set_title('Test Confusion matrix')
plt.show()
```



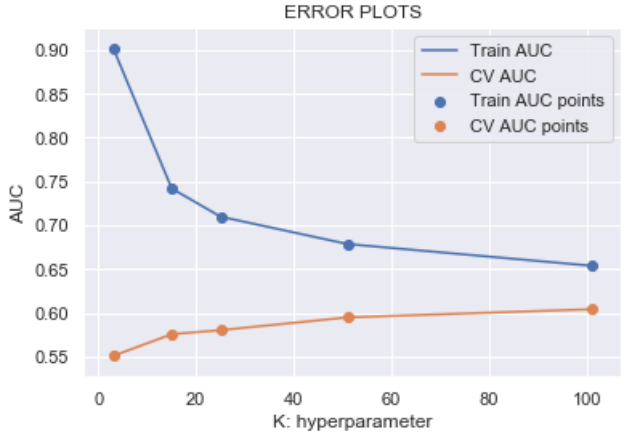
12.4. Applying KNN brute force on TFIDF W2V, SET 4

12.4.1. Finding The Best Hyperparameter "K"

In [187]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
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 = []
K = [3, 15, 25, 51, 101]
for i in tqdm(K):
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(Xtrain3, y_train)

    y_train_pred_tfidfw2v = batch_predict(neigh, Xtrain4)
    y_cv_pred_tfidfw2v = batch_predict(neigh, Xcv4)

    # 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_tfidfw2v))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred_tfidfw2v))

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

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

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

```
  0%|                                                                                           |
[00:00<?, ?it/s]
 20%|████████████████                                                                 | 1/5
[16:58<1:07:53, 1018.41s/it]
 40%|████████████████████████████████                                                   | 2/5
[33:49<50:48, 1016.32s/it]
 60%|████████████████████████████████████████████████                                   | 3/5 [47:4
<32:01, 960.73s/it]
 80%|████████████████████████████████████████████████████████████████                 | 4/5 [1:03:1
5<15:52, 952.97s/it]
100%|████████████████████████████████████████████████████████████████████████████████| 5/5
[1:18:39<00:00, 944.15s/it]
```



In [188]:

```
score_cv = [x for x in cv_auc]
optimal_K_cv = K[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_K_cv, '\n')
best_K_tfidfw2v = optimal_K_cv
print(best_K_tfidfw2v)
```

```
Maximum AUC score of cv is: 0.6042489689024103
Corresponding alpha value of cv is: 101

101
```

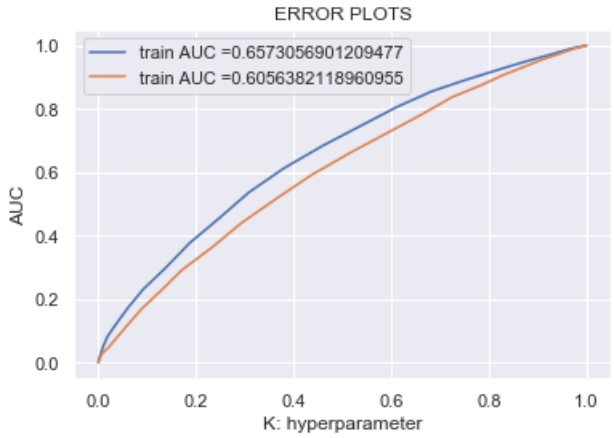
12.4.2. Testing the performance of the model on test data, plotting ROC Curves

In [145]:

```
# best_k_tfidf = 101
```

In [189]:

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


neigh = KNeighborsClassifier(n_neighbors=best_K_tfidfw2v, n_jobs=-1)
neigh.fit(Xtrain4, 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_tfidfw2v = batch_predict(neigh, Xtrain4)
y_test_pred_tfidfw2v = batch_predict(neigh, Xtest4)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_tfidfw2v)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_tfidfw2v)

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



12.4.3. Building Confusion Matrix

In [190]:

```
# 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 [191]:

```
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_tfidfw2v, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_tfidfw2v, best_t)))
```

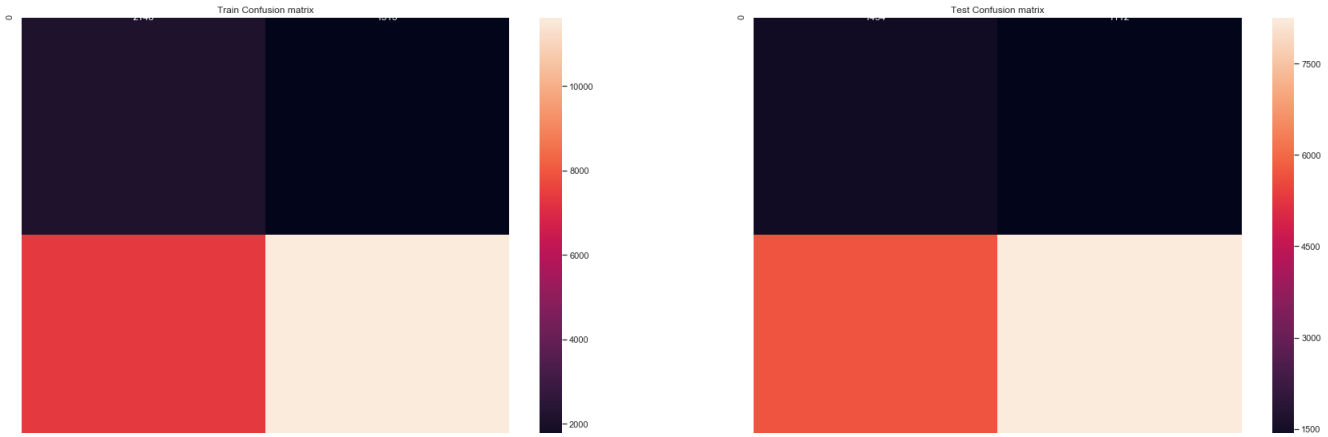
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.37977002880032895 for threshold 0.851
Train confusion matrix
[[ 2148  1315]
 [ 7360 11622]]
Test confusion matrix
[[1434 1112]
 [5699 8255]]
```

In [192]:

```
confusion_matrix_train_tfidfw2v = pd.DataFrame(confusion_matrix(y_train,
predict_with_best_t(y_train_pred_tfidfw2v, best_t)))
confusion_matrix_test_tfidfw2v = pd.DataFrame(confusion_matrix(y_test,
predict_with_best_t(y_test_pred_tfidfw2v, best_t)))
```

In [193]:

```
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_tfidfw2v,annot = True ,ax = axes[0],fmt='g')
sns.heatmap(confusion_matrix_test_tfidfw2v,annot = True , ax = axes[1],fmt ='g')
axes[0].set_title('Train Confusion matrix')
axes[1].set_title('Test Confusion matrix')
plt.show()
```



13. Feature selection with `SelectKBest`

In [ ]:

```
# 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 [160]:

```
from sklearn.feature_selection import SelectKBest, chi2 ,f_classif
selector = SelectKBest(score_func = chi2, k=2000)

X_train_new_2 = selector.fit_transform(Xtrain2,y_train)
X_test_new_2 = selector.transform(Xtest2)
X_cv_new_2 = selector.transform(Xcv2)
```

In [159]:

```
# print(X_best_kfeatures_set2[:5])
```

In [163]:

```
selector.get_support(indices=True)
```

Out[163]:

```
array([    0,     1,     4, ..., 10029, 10030, 10032], dtype=int64)
```

In [173]:

```
print(X_train_new_2.shape)
print(X_test_new_2.shape)
print(X_cv_new_2.shape)
```

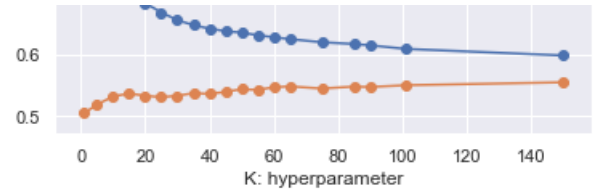
```
(22445, 2000)
(16500, 2000)
(11055, 2000)
```

In [178]:

```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
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 = []
K = [1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50 , 55, 60, 65, 75, 85, 90,101,150]
for i in tqdm(K):
```

```
    neigh = KNeighborsClassifier(n_neighbors=i, n_jobs=-1)
    neigh.fit(X_train_new_2, y_train)

    y_train_pred_new = batch_predict(neigh, X_train_new_2)
    y_cv_pred_new = batch_predict(neigh, X_cv_new_2)

    # 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_new))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred_new))

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

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

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

```
  0%|                                                                                           | 0
[00:00<?, ?it/s]
  5%|████                                                                                 | 1/19 [00:
14:37, 48.73s/it]
 11%|████████                                                                             | 2/19 [01:
14:20, 50.60s/it]
 16%|████████████                                                                         | 3/19
[02:39<13:52, 52.05s/it]
 21%|████████████████                                                                     | 4/19
[03:33<13:12, 52.85s/it]
 26%|████████████████████                                                                 | 5/19 [04:
<13:18, 57.02s/it]
 32%|████████████████████████                                                             | 6/19 [05:
<12:42, 58.65s/it]
 37%|████████████████████████████                                                         | 7/19 [06:
<11:29, 57.44s/it]
 42%|████████████████████████████████                                                     | 8/19
[07:33<10:25, 56.91s/it]
 47%|████████████████████████████████████                                                 | 9/19 [08:
7<09:21, 56.17s/it]
 53%|████████████████████████████████████████                                            | 10/19 [09:
2<08:21, 55.68s/it]
 58%|████████████████████████████████████████████                                        | 11/19 [10:
7<07:23, 55.41s/it]
 63%|████████████████████████████████████████████████                                    | 12/19 [11:
2<06:28, 55.49s/it]
 68%|████████████████████████████████████████████████████                                | 13/19
[12:07<05:31, 55.20s/it]
 74%|████████████████████████████████████████████████████████                            | 14/19 [13:
02<04:35, 55.12s/it]
 79%|████████████████████████████████████████████████████████████                        | 15/19 [13:
57<03:40, 55.02s/it]
 84%|████████████████████████████████████████████████████████████████                    | 16/19 [14:
51<02:44, 54.94s/it]
 89%|████████████████████████████████████████████████████████████████████                | 17/19 [15:
47<01:50, 55.29s/it]
 95%|████████████████████████████████████████████████████████████████████████            | 18/19
[16:42<00:55, 55.14s/it]
100%|████████████████████████████████████████████████████████████████████████████████| 19/19
[17:37<00:00, 55.15s/it]
```



In [179]:

```
score_cv = [x for x in cv_auc]
optimal_K_cv = K[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_K_cv, '\n')
best_K_new = optimal_K_cv
print(best_K_new)
```

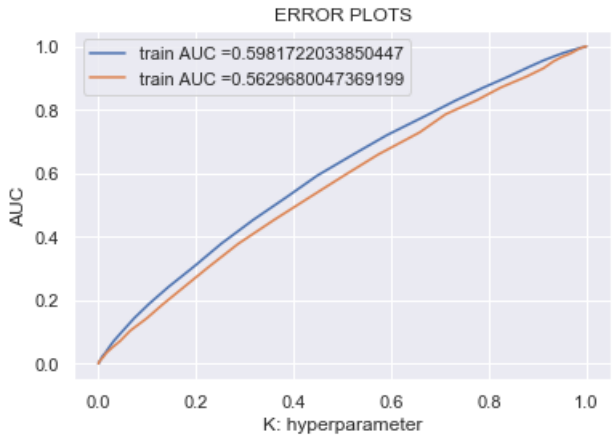
```
Maximum AUC score of cv is: 0.5548632678344598
Corresponding alpha value of cv is: 150

150
```

In [180]:

```
# best_k_new = 50
```

In [182]:

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


neigh = KNeighborsClassifier(n_neighbors = best_K_new, n_jobs=-1)
neigh.fit(X_train_new_2, 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_new = batch_predict(neigh,X_train_new_2)
y_test_pred_new = batch_predict(neigh, X_test_new_2)

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred_new)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred_new)

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



In [183]:

```
# 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 [184]:

```
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_new, best_t)))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict_with_best_t(y_test_pred_new, best_t)))
```

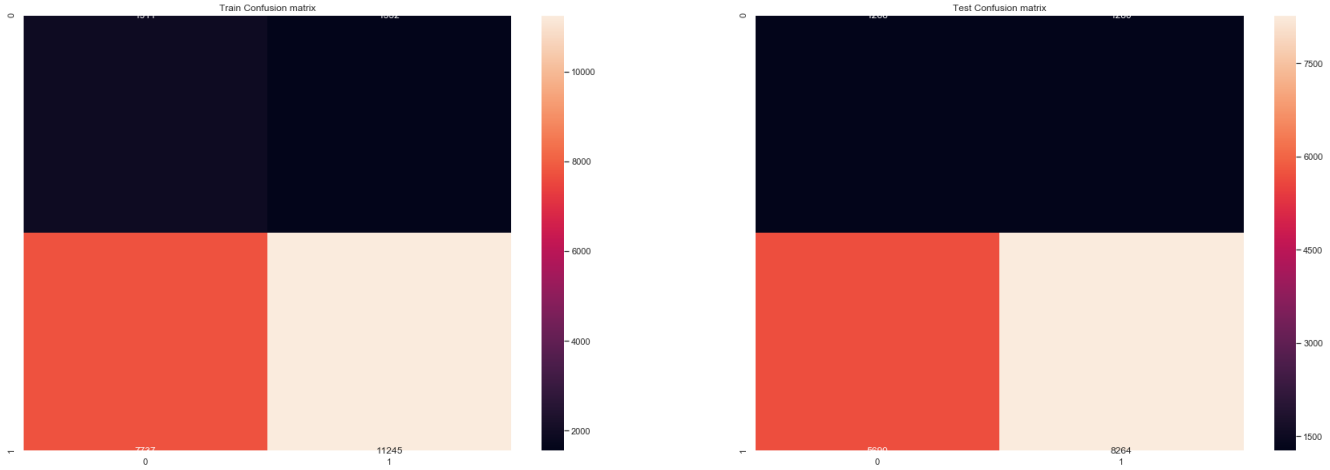
```
====================================================================================================

the maximum value of tpr*(1-fpr) 0.3269081035568052 for threshold 0.84
Train confusion matrix
[[ 1911  1552]
 [ 7737 11245]]
Test confusion matrix
[[1266 1280]
 [5690 8264]]
```

In [185]:

```
confusion_matrix_train_scores = pd.DataFrame(confusion_matrix(y_train,
predict_with_best_t(y_train_pred_new, best_t)))
confusion_matrix_test_scores = pd.DataFrame(confusion_matrix(y_test,
predict_with_best_t(y_test_pred_new, best_t)))
```

In [186]:

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



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(["BOW", "Brute", 101, 0.6673,0.6335])
x.add_row(["TFIDF", "Brute", 101, 0.6294,0.5808])
x.add_row(["W2V", "Brute", 101, 0.6545,0.5900])
x.add_row(["TFIDF W2V","Brute", 101, 0.6624,0.5979])
x.add_row(["TFIDF 2000 features", "Brute", 40, 0.6500,0.4972])
```

In [156]:

```
print(x)
```

```
+---------------------+-------+-----------------+------------+-----------+
|      Vectorizer     | Model | Hyper parameter | Train AUC | Test AUC |
+---------------------+-------+-----------------+------------+-----------+
|         BOW         | Brute |       101       |   0.6673   |  0.6335  |
|        TFIDF        | Brute |       101       |   0.6294   |  0.5808  |
|         W2V         | Brute |       101       |   0.6545   |   0.59   |
|      TFIDF W2V      | Brute |       101       |   0.6624   |  0.5979  |
| TFIDF 2000 features | Brute |        40       |    0.65    |  0.4972  |
+---------------------+-------+-----------------+------------+-----------+
```

So, we conclude that our SET1 BOW auc score is higher than all and hence it works better

In [ ]: